MASTER'S THESIS DATA SCIENCE

RADBOUD UNIVERSITY NIJMEGEN

The temporal WTA circuit: a novel neuromorphic approach for learning to distinguish time-varying stimuli

Author:
Otto VAN DER HIMST
s4493591

Supervisor:
David VAN LEEUWEN

Second reader:
Louis TEN BOSCH

December 2023

***Abstract*—With the limit of Moore's law drawing near, alternative methods of computation become increasingly relevant. One such alternative is that provided by neuromorphic architectures, which perform computations in a brain-inspired fashion. Neuromorphic properties such as sparsity and locality allow neuromorphic computers to function with orders of magnitude smaller energy and latency costs than their von Neumann counterparts. However, in order to utilize the potential of neuromorphic hardware, algorithms must be developed that themselves operate according to neuromorphic principles. One such algorithm is the Winner-Take-All (WTA) circuit, a neuromorphic SNN algorithm that via plasticity mechanisms learns — unsupervised, online, and in real time — to distinguish between spike-encoded stimuli. A limitation of this algorithm is that it can only distinguish between static (rate-coded) spike patterns, but not time-varying ones. In this work we present the temporal WTA circuit. Unlike the traditional (static) WTA circuit, the temporal WTA circuit is able to distinguish between time-varying spike patterns. The experiments show, first of all, that the temporal WTA circuit is indeed capable of learning to distinguish between time-varying spiking patterns, where the static WTA circuit is not. Secondly, the experiments show how temporal WTA circuit can be combined into a network, where separate layers operate at different timescales, thereby allowing the network to process temporally larger stimuli. Thirdly, using the TIDIGITS dataset, the experiments show how a two-layer temporal WTA network can learn to perfectly remember utterances of a single speaker. Finally, the experiments show how the two-layer temporal WTA network is capable of learning to distinguish between the spoken digits.**

***Index Terms*—neuromorphic computing, Spiking Neural Network (SNN), temporal Winner-Take-All (WTA) circuit, time-varying pattern recognition, speech recognition, real-time online learning, unsupervised learning**

I. Introduction

COMPUTERS are ubiquitous in modern life, fulfilling countless roles both large and small. The vast majority of computers follow the von Neumann architecture, which – as predicted by Moore's Law [1] – have seen an exponential growth in computational power over the past decades. However, as we approach the limit of Moore's Law, qualitative innovations become necessary to improve the capabilities of computers, including the development of alternative computing architectures.

One prominent alternative to the von Neumann architecture, are neuromorphic architectures. Neuromorphic architectures draw inspiration from the biological brain, and attempt to process information in a similar manner. There is as of yet no consensus about what it means for an architecture to be neuromorphic, with such architectures varying greatly in the degree to which they adhere to neurobiological principles. Even so, there is enough consensus about the general principles involved for it to have become a distinct field of research, having been projected to make up 20% of all Artificial Intelligence (AI) computing and sensing by 2035 [2]. In order to understand the benefits that neuromorphic computing can offer over von Neumann computing, comparisons follow with respect to three domains: hardware, algorithm, and application.

A. Hardware

1) The von Neumann architecture: The primary distinction within the von Neumann architecture is that between the centralized control that resides in the Central Processing Unit (CPU), and a separate storage area which functions as the primary memory. Within the primary memory, instructions and data are stored [3]. This information is fetched and executed sequentially by the CPU, which in turn writes new information back to the primary memory. The constant flow of information between the CPU and the primary memory can be a significant source of delay and comes with high energy costs, this is referred to as the von Neumann bottleneck [4].

2) The neuromorphic architecture: The neuromorphic architecture is less clearly defined. General consensus is that neuromorphic designs draw inspiration from the biological brain, though exactly which aspects it is inspired by, and to what extent it mimics them, varies greatly. But what is it about the biological brain that inspires?

The human brain consists of many billions of neurons and many trillions of connections [5], arranged into many interconnected biological Spiking Neural Networks (SNNs). Spikes of information are emitted by neurons and travel via axons, synapses, and dendrites to excite or inhibit a selection of thereby connected neurons. The true complexity of the brain is much more intricate, and is far from understood in its entirety. This elementary description, however, is sufficient to touch on several properties of the biological brain that serve as inspiration for neuromorphic computing.

The internal state of 'brain constituents' (neurons, synapses, ...) changes smoothly over (*continuous*) time, in an *analogue* fashion. For example, the change in the membrane potential of a neuron at a given time, is dependent on its membrane potential and on other properties of the neuron at that time (*temporal locality*): typically a neuron will drift tend towards some resting potential. Additionally, the neuron's membrane potential may be perturbed in an *event-based manner* by signals arriving via *spatially local* connections. These inherent properties of the biological brain allow a number of additional properties to emerge.

First of all, from the property of (spatial and temporal) locality follows the *co-location of memory and computation*: inherent to the state and properties of brain constituents is the information required to perform specific computations. Secondly, by depending solely on local and event-based computation each constituent is able to function in *parallel* to the others. Thirdly, if a component is at its natural resting state (at equilibrium), then it can remain dormant until it is perturbed by an incoming signal. At any given time, the large majority of neurons in the biological brain are in this sense dormant, yielding a pattern of extremely *sparse* computation.

We can observe two more properties of the biological brain that are often taken as inspiration for neuromorphic architectures. Over the course of evolution many different types of biological brains have emerged. Biological brains emerging in later stages of evolution often contain many brain regions that are clearly identifiable in earlier stages of evolution, but have new brain regions working in concert with these. This indicates that the biological brain is an inherently *scalable* system [6]. Finally, there is the observation that the nature of molecular and cellular processes that underlie the dynamics of the brain are inherently *stochastic*, and that thereby so too is

the behaviour of the biological brain [7].

3) Von Neumann vs Neuromorphic: Several benefits are associated with neuromorphic architectures when compared to the Von Neumann architecture, as well as several drawbacks. First of all, on certain applications, neuromorphic computers can achieve *energy-efficiency* several orders of magnitude beyond what von Neumann computers are capable of [8]–[10]. One significant source of this contrast is the elimination of the von Neumann bottleneck, the co-location of memory and computation removing the need for the constant and expensive transfer of information between CPU and memory. Another is the ability of a neuromorphic computer to perform truly sparse computations, allowing it to expend energy only when and where it needs to.

Secondly, the elimination of the von Neumann bottleneck, and the parallel and event-based nature of the neuromorphic architecture, yields the potential for extremely *low-latency* computation [10], [11]. While in the von Neumann architecture response to a stimuli is limited to occur at the next clock cycle, event-based systems begin processing a stimulus as soon as it arrives.

While the aforementioned benefits have thus far received most attention in neuromorphic research, several additional benefits have been proposed, and more may be found in time [9]. For one, it has been suggested that the stochastic nature of a neuromorphic architecture can be used to exploit intrinsic high-level randomness in existing or emerging device technology to efficiently perform computations [12]. For another, analogue neuromorphic architectures allow for types of computation not possible under the digital von Neumann architecture, such as the possibility to represent continuous values and operate in the domain of continuous time [13].

Despite the wide range of benefits that neuromorphic architectures offer, there are fronts on which the von Neumann architecture is more suitable. For one, the deterministic computation offered by the von Neumann architecture will remain desirable, or even necessary, for many applications. Likewise, the ability to access specific information at specific times in a global memory bank is, and will remain, desirable for many tasks.

Furthermore, at least for some years to come, the von Neumann architecture has a significant head start on neuromorphic architectures. Nearly all computers used in everyday life stem from the von Neumann architecture [14], and peripherals and protocols have been designed and optimized over many years to work with such systems. Consider for example the creation of a neuromorphic processor that matches state-of-the-art von Neumann processors at some task, but operates with several orders of magnitude greater energy-efficiency. While the neuromorphic processor may seem vastly superior, there are several obstacles that diminish its advantage. First, it might not be straightforward to smoothly interface the neuromorphic processor with the appropriate peripherals. Second, the energy-efficiency benefits of the neuromorphic processor may be diminished by the fact that the peripherals also consume a significant amount of energy: if the peripherals consume 10% of the total system's energy, then improving the energy-efficiency of the processor can only ever yield improvements up to a factor of ten. The development of all-round neuromorphic systems is therefore of vital importance if neuromorphic computers are to thrive; a process that will take significant time and effort.

B. Algorithm

1) Algorithmic differences: In order to utilize the strength of each computing architecture, algorithms must be employed that match their strengths. A von Neumann computer will not benefit from an algorithm requiring access only to local information, whereas a neuromorphic computer typically will. The same neuromorphic computer, however, may not be able to run an algorithm that requires access to global information. As an example, graph algorithms are considered a promising family of algorithms for neuromorphic architectures [15]. Graph nodes can readily be modelled by neurons, and weighted edges by synapses. Alternatively a neuromorphic design may abstract away from SNNs entirely and directly emulate graph structures. Most prominently however, neuromorphic architectures show promise with respect to AI and machine learning algorithms.

2) Artificial Intelligence: The field of AI has grown substantially in recent years. Two decades ago the achievements of AI were trivial when compared to human intelligence, while today AI can tackle – and even sometimes outperform humans on – a wide range of complex tasks [16], [17]. The rapid rise of AI in recent years was driven by the exponentially increasing computational power of hardware, the development of specialized hardware such as GPUs and TPUs, the growing AI community both on a an amateur and a professional level, and – on an algorithmic level – the success of backpropagation and stochastic gradient descent in the field of deep learning [17], [18]. Deep learning algorithms now thrive on von Neumann computers, but – due to their non-local and sequential nature – are not straightforwardly suitable for neuromorphic computers.

Instead of the Artificial Neural Networks (ANNs) that underlie deep learning methods, Spiking Neural Networks (SNNs) are more naturally suited for neuromorphic architectures. In fact, neuromorphic hardware often directly emulates the structure of SNNs. While – like neuromorphic computers themselves – SNNs can be implemented in many ways, they typically consist of neurons connected into a network (e.g., via axons, synapses, and dendrites), where (typically binary) spike signals are sent between connected neurons. Information is represented by patterns of spiking neurons. This yields a distinguishing characteristic of SNNs, namely that the manner in which information is encoded includes a notion of time: information can be encoded in the relative timing of spiking neurons, and learned information can be encoded in connection delays. SNNs are uniquely suitable for exploiting the neuromorphic characteristics mentioned in section I-A – unsurprisingly, since these characteristics are inspired by biological SNNs.

3) Learning: One of the major challenges for neuromorphic AI is learning. While in methods such as backpropagation and gradient descent have seen great success within the field of deep learning, these are not straightforward to apply in SNNs.

One obstacle in this regard is the fact that SNNs (typically) communicate via binary spikes, which are not differentiable. Another is the fact that these methods require information that is non-local in space and time. Still, approaches exist to train SNNs using these methods.

One such approach is to first train an ANN using Back-Propagation Through Time (BPTT), mapping it to a SNN afterwards. This approach has yielded near state-of-the-art performance, with the potential for considerably reduced energy costs [9], [19]–[21]. However, reduced accuracy is not out of the question. Furthermore, the training phase utilizes none of the benefits that neuromorphic approaches have to offer, and additional costs are incurred by the mapping from ANN to SNN.

An alternative approach is to approximate well-developed machine learning methods, such that learning can occur directly – on-chip – in the SNN [22]–[25]. One prominent example of such a method is e-prop [25], which approximates BPTT. E-prop draws inspiration from two pieces of neuroscientific knowledge. First of all, from the knowledge that the biological brain maintains (locally) temporary records of events in the form of so-called eligibility traces. Secondly, from the knowledge that the biological brain sends top-down learning signals in order to inform specific populations of neurons of behavioural results. Using such mechanisms, e-prop is capable of approximating BPTT and – though it learns slower than BPTT – e-prop facilitates online and on-chip learning in Recurrent SSNs (RSNNs).

The aforementioned learning methods have the benefit of using and recycling knowledge from the well-developed and successful field of deep learning. However, as noted by [9], limiting ourselves to algorithms designed for ANNs may also limit what will be achieved by SNNs. While adhering solely to such strategies, neuromorphic approaches may yield improved performance in terms of energy and latency, but not in terms of measures such as accuracy. In addition to repurposing existing methods, new learning methods should be (and are being) developed to exploit properties that are inherent SNNs.

Again, we can look towards the biological brain for inspiration. In the brain it can be observed that learning occurs via local plasticity mechanisms that adapt properties of neurons, synapses, and other brain components. Rules used to guide this adaptation, both in neuroscience and in neuromorphic hardware, generally follow the shape of Three Factor (3F) rules. According to such a rule, synaptic change is determined by (1) pre-synaptic and (2) post-synaptic states, and (3) by a post-synaptic modulation term which can represent signals such as error (supervised), reward (reinforcement), and surprise (unsupervised) [10]. The local and event-based nature of 3F-based methods makes them uniquely suitable for energy-efficient, continual, on-chip learning.

C. Applications

There is a wide variety of applications that can benefit from neuromorphic technology. In the short term, the most obvious applications are those that require low-latency or low-energy computation. Examples of such applications include robotics, autonomous vehicles and drones, and all kinds of edge applications [9], [10]. Within all these domains is the desire to have the device in question function on battery for extended periods of time, for it to be able to respond quickly to a variety of sensory stimuli, and for it to continuously learn and adapt to its environment. Neuromorphic technology is uniquely suited to fulfil these desires.

One commonality between the aforementioned applications, is that much of their functioning will depend on processing and interpreting sensory stimuli. In particular, many environments will require the processing of time-varying signals. A drone for example is faced with rapidly changing visual scenes, and a voice assistant has to understand and react to speech smoothly in real time. The event-based nature of neuromorphic approaches, and their ability to extract information from temporal variations in signals, makes them uniquely suited to process such time-varying stimuli.

This work focuses on the development of a neuromorphic algorithm suitable for the aforementioned applications. Specifically, this work introduces a novel SNN algorithm that utilizes plasticity rules in order to learn — unsupervised, online, and in real time — to distinguish between time-varying (spike) signals. We will argue that, in principle, the nature of the algorithm allows it to exploit the neuromorphic principles of locality, sparsity, parallelism, stochasticity, and scalability, and allows for event-based, continuous, and analogue computation. The remainder of this work is structured as follows. Section II provides insight into related work. Section III provides a formal description of the algorithm. Section IV evaluates the behaviour and performance of the algorithm in a variety of experiments. Section V provides a general discussion of the algorithm, the experimental results, and directions for future work. Finally, section VI ends this work with a concluding statement.

II. Related Work

A. Tempotron

Gütig et al. [26] proposed a biologically plausible supervised classifier called the tempotron. The tempotron is a SNN that is able to extract information from the spatiotemporal structure of spike trains. In particular, this includes the ability to classify time-varying (in addition to rate-coded) spiking patterns. The tempotron consists of a layer of Leaky Integrate-and-Fire (LIF) neurons, which is fully connected to an input layer that encodes stimuli as spiking patterns. Each LIF neuron is associated with a single label, a spike from the neuron – elicited when its membrane potential exceeds its firing threshold – indicating classification of the neuron's corresponding label. The output spike of the tempotron is not fixed in time, rather allowing event-based classification.

When a neuron makes a mistake, which can take the form of a false positive or a false negative, learning occurs via local plasticity rules, steered by a binary error signal. If the neuron responds to the wrong label (false positive), then the error signal elicits a decrease in certain synaptic weights, if it does not respond to the right label (false negative), then the error signal elicits an increase in certain synaptic weights. The

degree of change of each synaptic weight is dependent on its specific contribution to the error.

Though for the most part the tempotron offers a biologically plausible supervised classification method, the authors note two limitations. First of all, it is not clear how the supervision signal arrives at the site of plasticity. Secondly, it is not clear how the signal is translated into appropriate synaptic changes. The supervisory signal only needs to contain information about the polarity of the synaptic change: decrease for false positive and increase for false negative. Ideally then, the synapses themselves locally maintain the information necessary to determine the strength of the synaptic changes. The authors do not provide concrete solutions for these problems, though they briefly propose that neuromodulatory pathways can be recruited to activate either Long-Term Potentiation (LTP) or Long-Term Depression (LTD) after an error, and that eligibility traces might be maintained in order to determine the appropriate amount of synaptic change when a supervision signal arrives.

Finally, the authors note that the method is restricted to stimuli that fit the relatively short integration window of the tempotron, and that additional memory mechanisms are required to solve more demanding tasks. In following sections II-B and II-C we will see the tempotron being used in such a fashion.

B. Liquid State Machine

The Liquid State Machine (LSM) is one of the more prominent neuromorphic paradigms for real-time processing of time-varying signals [27], [28]. The LSM can be separated into two major components. On the one hand, a 'liquid' that can be perturbed by external stimuli. On the other hand, a memoryless readout unit that interprets the present state of the liquid.

The form of the liquid can vary greatly, from a literal liquid (e.g., water) where ripples on the surface temporarily maintain information about past perturbations, to a SNN where input spikes temporarily resonate through a network of recurrently connected neurons. In general though, it is desirable that the liquid remembers successive inputs via some dynamic, non-linear, and high-dimensional representation. The liquid is the more complex part of the LSM. The readout unit is typically kept relatively uncomplicated, and often takes the form of a simple linear regression classifier.

A key characteristic of the LSM is that it can function without any learning being done by the relatively complex liquid. Instead, the liquid is often initialized at random according to simple heuristics, and remains unchanged after [29]. Under the right circumstances, it is then able to non-linearly separate the input and maintain information about recent perturbations. A linear classifier is often sufficient to learn (in a supervised manner) how to interpret different liquid states. It has been noted, however, that tuning of the liquid through some type of meta-learning can dramatically improve performance [29].

Various properties of the LSM make it well suited for neuromorphic implementation. For one, the entirety of a LSM (input, liquid, classifier) can take the form of a (sparsely firing, recurrent) SNN. Sensory inputs can be encoded via spike trains using event-based sensors [30]–[32], the liquid can be a sparsely and recurrently connected SNN, and the classifier can be a tempotron. In this form it is able — in real time — to (learn to) extract information from temporal variations in input signals. Furthermore, multiple classifiers can be taught to extract different information from the same liquid, allowing for parallel classification. As such, it has been noted to be an attractive SNN model for low-power edge computing [33].

Despite all this, it should be noted that the performance of LSM methods has not yet caught up to the state-of-the-art performances of BPTT-trained SNNs [33]. Also, it has been noted that by [29] that due to challenges connected to catastrophic forgetting, LSM methods are difficult to apply in online learning settings. Finally, the LSM in general does not actually provide a solution for neuromorphic learning, which is a task relegated to other methods such as linear regression classifiers; instead the LSM just provides a type of short-term memory.

C. Self-Organizing Map

The Self-Organizing Map (SOM) [34] is an ANN capable of tuning itself to observed patterns via a process of unsupervised learning. It exhibits an interesting property that is also found in biological brains, namely that the sensitivity of groups of neurons becomes spatially organized to match some dimension of the data to which they are exposed. In other words, neighbouring neurons learn to become sensitive to similar input patterns. This is seen all over the human brain, particularly on a sensory level [34]. In the visual areas we find colour maps [35], in the auditory cortex we find tonotopic maps organized according to pitches of tones [36], and somatotopic maps follow a spatial representation of our body [37].

The SOM consists of a layer of artificial neurons with differing sensitivities to specific input stimuli. Through mechanisms of competition and collaboration, each neuron learns to become distinctly sensitive to a particular input pattern. When a stimulus is presented to the SOM, the neuron that is most sensitive to the stimulus responds, this neuron is referred to as the Best Matching Unit (BMU). Sensitivity is computed as the distance between a neuron's synaptic weight vector and the input vector representing the stimulus. After determining the BMU, the synaptic weight vector of the BMU is updated to become closer to the input vector, and thus to become more sensitive to the stimulus at hand. Neurons adjacent to the BMU adapt in the same manner, but to a lesser degree, the degree to which a neuron does so being inversely proportional to its distance to the BMU. This unsupervised learning process eventually causes each neuron to become distinctly sensitive to specific patterns, with neighbouring neurons becoming sensitive to similar patterns, according to the distribution of the data.

Aiming to introduce a new neuromorphic approach for speech recognition, Wu et al. [38] proposed the SOM-SNN framework. The method utilizes the SOM in order to extract acoustic features from speech, learning to do so in an unsupervised manner. The method consists of three main components

organized in a pipeline. First, the speech signal – in the form of Mel-scaled filters – is encoded into spikes via latency coding, where the earlier arrival of spikes indicates a higher amplitude of the corresponding signal. Secondly, a single SOM layer is used to extract acoustic features from real-valued vectors that correspond to the spike times of the encoded speech signal. Thirdly, in sequential order according to their respective time frame, each BMU generates a spike, thereby generating a sparse spiking pattern to be interpreted by a tempotron layer. The tempotron layer is trained in a supervised manner to classify these spiking patterns.

The authors report state-of-the-art accuracy of 97.60% on the TIDIGITS dataset. If the SOM layer is omitted from the pipeline, and the tempotron thus has to directly classify the encoded speech signal, the authors report an accuracy of 9.11% (chance). As the authors note, this contrast matches with similar work in the field of deep learning, where the extraction of discriminative features play an important role in ASR [39].

The work of [38] shows that the SOM is a suitable method extracting meaningful acoustic features in an unsupervised manner. From a neuromorphic point of view, however, the method has several weaknesses. First of all, the time-variation of the spike encoding is not used to encode the time-variation of the speech signal, instead it is used solely to encode the amplitude of the signal. Such an encoding is not suitable for real-time processing of speech. Secondly, the SOM is not a SNN, but an ANN. Input to the SOM is given as a real-valued vector corresponding to spike times, and the BMUs do not generate a spike pattern in an event-based manner. Finally, in order to find the BMU, non-local comparisons need to be made to find the shortest distance between synaptic weight vectors and the input vector.

D. WTA circuits and networks

Another neuromorphic approach is driven by two pieces of neuroscientific evidence. First, the observation that the human brain appears to process information in a Bayesian manner. And second, the ubiquity of so-called Winner-Take-All circuits in the human brain. Nessler et al. [40] propose that WTA circuits are capable of performing object classification via Bayesian computations.

A WTA circuit is a simple SNN consisting of a single layer of neurons that receive spike input via synaptic connections with other neurons (such as sensory neurons). Each neuron in the WTA layer is excited by spikes coming in from pre-synaptic neurons, weighed by corresponding synaptic weights. The spiking probability of a WTA neuron increases exponentially as a function of its membrane potential. When a WTA neurons spikes, a strong inhibitory spike signal is sent to all WTA neurons, reducing the chance that other neurons respond to the same signal. Additionally, following a spike the synaptic weights of the spiking WTA neuron are updated via plasticity dynamics: synaptic weights which recently saw pre-synaptic spiking activity are strengthened, those that did not are weakened. Over time, these dynamics cause each WTA neuron to become sensitive to a distinct pre-synaptic spiking pattern.

Nessler et al. [40] formally compare this process to the Expectation Maximization (EM) algorithm. The authors show that the spiking of a WTA neuron can be viewed as the expectation-step. In this view, the spike can be interpreted as a sample from a posterior over hidden causes, which is encoded in the synaptic weights of the WTA neurons. Following a spike, the corresponding STDP weight update can be seen as the maximization step, moving the synaptic weights closer to the observed distribution.

Nessler et al. [41] illustrate this process using the MNIST dataset of handwritten digits. In their experiments they present rate-encoded MNIST digits to a WTA circuit consisting of 100 neurons. Their results show that merely by unsupervised exposure to these spike patterns, the WTA neurons become distinctly sensitive to varying handwritten digits, achieving a classification accuracy of 80.14%. Guo et al. [42] extend this method by combining a multitude of WTA circuits into a two-layer hierarchical WTA network. The first layer consists of sixteen 15-neuron WTA circuits which each observe a separate 7×7 pixel segment of rate-encoded MNIST images. The second layer consists of a single 100-neuron WTA circuit which receives input from all 16×15 WTA neurons in the first layer. Their method achieves an accuracy of 84.89% on the same benchmark. Van der Himst et al. [43] replicate the method of [42] and show that multiple hierarchical WTA networks can be combined to process multiple stimuli. They further show that feedback mechanisms can facilitate communication between otherwise independent parts of a network, and that such mechanisms can improve learning.

While WTA networks have yet to be applied to more complex problems than the MNIST, they are extremely well suited for neuromorphic implementation. Though simulations on von Neumann computers may require the model to be implemented in discrete time, the original WTA circuit model of [40] was conceived to operate within a continuous time domain, and is suitable for analogue implementation (as evidenced by the presence of WTA circuits in the human brain). Furthermore, all operations can be realized in a local, event-based, and parallel manner, eliminating the problem of the von Neumann bottleneck. The lateral inhibition mechanisms fundamental to WTA circuits guarantee sparse spiking activity. The stochastic nature of a WTA neuron's firing matches the stochastic nature of biological brains. And the work of [42] and [43] show that the method is naturally scalable, and can benefit from this scalability. Finally, a fundamental feature of WTA circuits is their capacity for online, real-time, and unsupervised learning.

Despite all these desirable neuromorphic properties, the method has a significant limitation. Thus far it has only been applied to static (rate-coded) spike patterns. If it is to be applied in real-world settings then this will not suffice. In the real world visual scenes are rarely stationary, and many types of signals — such as speech — are characterized by temporal variation. In this work, a novel algorithm is proposed that enables WTA circuits to learn to distinguish between time-varying spike patterns, whilst keeping intact the aforementioned neuromorphic characteristics of WTA circuits. We call this algorithm the temporal WTA circuit, and contrast it against the traditional (static) WTA circuit.

III. Method

In the following we provide a formal WTA circuit definition, including a description of its dynamics. First, a generic definition will be given that fits with all the aforementioned works on WTA circuits [40]–[43]. Second, specific properties and dynamics of WTA circuits, as implemented in this work, will be defined. Finally, we introduce novel properties and dynamics that allow a WTA circuit to (learn to) distinguish between time-varying (rather than only static) spike patterns. Table I provides an overview of the notation used in this work. Figures 1 and 2 visualize the dynamics of both static and temporal WTA circuits.

A. Static WTA circuit definition

In essence, a WTA circuit consists of four sets of components. First of all, it consists of a collection of K WTA neurons $\boldsymbol{z} = \{z_1, ..., z_K\}$. Secondly, it consists a collection of N input neurons $\boldsymbol{y} = \{y_1, ..., y_N\}$. Thirdly, consists of a collection of $K \cdot N$ connections $\boldsymbol{c} = \{c_{kn} | k \in \{1, ..., K\}, n \in \{1, ..., N\}\}$ that allow spikes to travel from neurons $\boldsymbol{y}$ to neurons $\boldsymbol{z}$. Finally, a collection of (here undefined) connections that, following a spike from a neuron z_k, transport an inhibitory signal $I(t)$ to all neurons $\boldsymbol{z}$. Each set of components has several properties that determine the overall behaviour of the WTA circuit.

Within the general WTA circuit definition, the properties of input neurons $\boldsymbol{y}$ are largely undefined. It is required that neurons $\boldsymbol{y}$ produce some spike pattern, but the dynamics behind this change between applications. Neurons $\boldsymbol{y}$ may represent retinal neurons encoding visual stimuli, or may represent neurons $\boldsymbol{z}'$ of a different WTA circuit [42], [43], or may represent something else entirely. In previous works stimuli were encoded via Poisson spike trains [40]–[43]. In these works, the spiking probability of each input neuron is dependent on some static stimulus, and remains stable for the duration that a stimulus is presented. We refer to these types of encoding and to the corresponding WTA circuits, respectively as static encodings and static WTA circuits.

Connections $\boldsymbol{c}$ allow spikes to travel from neurons $\boldsymbol{y}$ to neurons $\boldsymbol{z}$. The connections are weighed by scalar weights $\boldsymbol{w}(t) = \{w_{kn}(t) | k \in \{1, ..., K\}, n \in \{1, ..., N\}\}$, such that a spike emerging from neuron y_n excites neuron z_k by an amount equal to w_{kn}. Weights $\boldsymbol{w}(t)$ evolve over time according to plasticity dynamics. This learning is triggered when a neuron z_k spikes, causing connection weights w_k to be updated. Generally, this entails that a connection weight w_{kn} is increased when y_n produced a spike within some time window before the spike from z_k, and is decreased otherwise.

Each neuron z_k has a scalar membrane potential $\mu_k(t)$ that changes over time. Specifically, when a neuron y_n spikes, connection c_{kn} weighs and transports the spike such that membrane potential $\mu_k(t)$ of neuron z_k is perturbed by an amount equal to connection weight w_{kn}. Whether or not a neuron z_k spikes at time t is dependent on its membrane potential at that time. We use $\phi_n(t)$ and $\zeta_k(t)$ to denote whether a neuron y_n and a neuron z_k spiked at time t:

$$\phi_n(t) = \begin{cases} 1 & \text{if neuron } y_n \text{ spiked at time } t \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

$$\zeta_k(t) = \begin{cases} 1 & \text{if neuron } z_k \text{ spiked at time } t \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

$$\zeta(t) = \begin{cases} 1 & \text{if any neuron in } \boldsymbol{z} \text{ spiked at time } t \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

We further use $\bar{\phi}_n(t)$, $\bar{\zeta}_k(t)$, and $\bar{\zeta}(t)$ respectively to denote the inverse of these functions.

This definition of a WTA circuit is a generic one that fits with the definitions used in earlier works [40]–[43]. When implementing a WTA circuit however, a more specific definition is needed, and choices have to be made. For example, the above definition does not specify whether the model operates in continuous or in discrete time, nor does not specify how quickly a spike travels from a neuron y_n to a neuron z_k, nor does it specify exactly how plasticity mechanisms cause the evolution of weights $\boldsymbol{w}$. In addition to these ambiguities, the model can be increased in complexity by adding additional properties and dynamics. The following provides a complete WTA circuit definition with details that are specific to this work.

B. Static WTA circuit implementation

The previous section provides a generic definition of static WTA circuits. This section describes the exact static WTA circuit properties and dynamics used in this work. Note that a number of the selected properties and dynamics violate neuromorphic properties. These choices were made because of practical considerations and limitations. On the one hand, because there is the desire to focus on essential parts of the method, and thus to not introduce additional variables where it can be avoided. On the other hand, because there is the fact that our WTA circuit implementation concerns simulation on a von Neumann computer, and thus can but approximate certain neuromorphic properties. The neuromorphic limitations imposed by these choices do not, however, extend to the method in general. Section V addresses this in detail.

This work adopts a discrete time model, where time t starts at 0 and is incremented in discrete steps of 1. Within a timestep a number of operations occur in sequential order. First, according to the input and the encoding scheme, it is decided which neurons in $\boldsymbol{y}$ spike. Second, these spikes are weighed and transported over connections $\boldsymbol{c}$ to neurons $\boldsymbol{z}$, where membrane potentials $\boldsymbol{\mu}$ are updated accordingly. Third, according to membrane potentials $\boldsymbol{\mu}$, it is determined which neurons in $\boldsymbol{z}$ spike. Fourth, if one or more neurons z_k spike (i.e., $\zeta(t) = 1$), then an inhibitory signal $I(t)$ instantly inhibits all membrane potentials $\boldsymbol{\mu}$. Finally, for each neuron z_k that spikes, connection weights $\boldsymbol{w}_k$ are updated according to plasticity mechanisms. At the onset of a new stimulus, variables that operate at a short timescale (e.g., membrane potentials, but not connection weights), are reset to their resting state, and time t is reset to 0.

Each stimulus is encoded by spike trains produced by input neurons $\boldsymbol{y}$. The manner in which these spike trains are produced varies between experiments and is addressed in detail in their respective sections. In general though, each stimulus

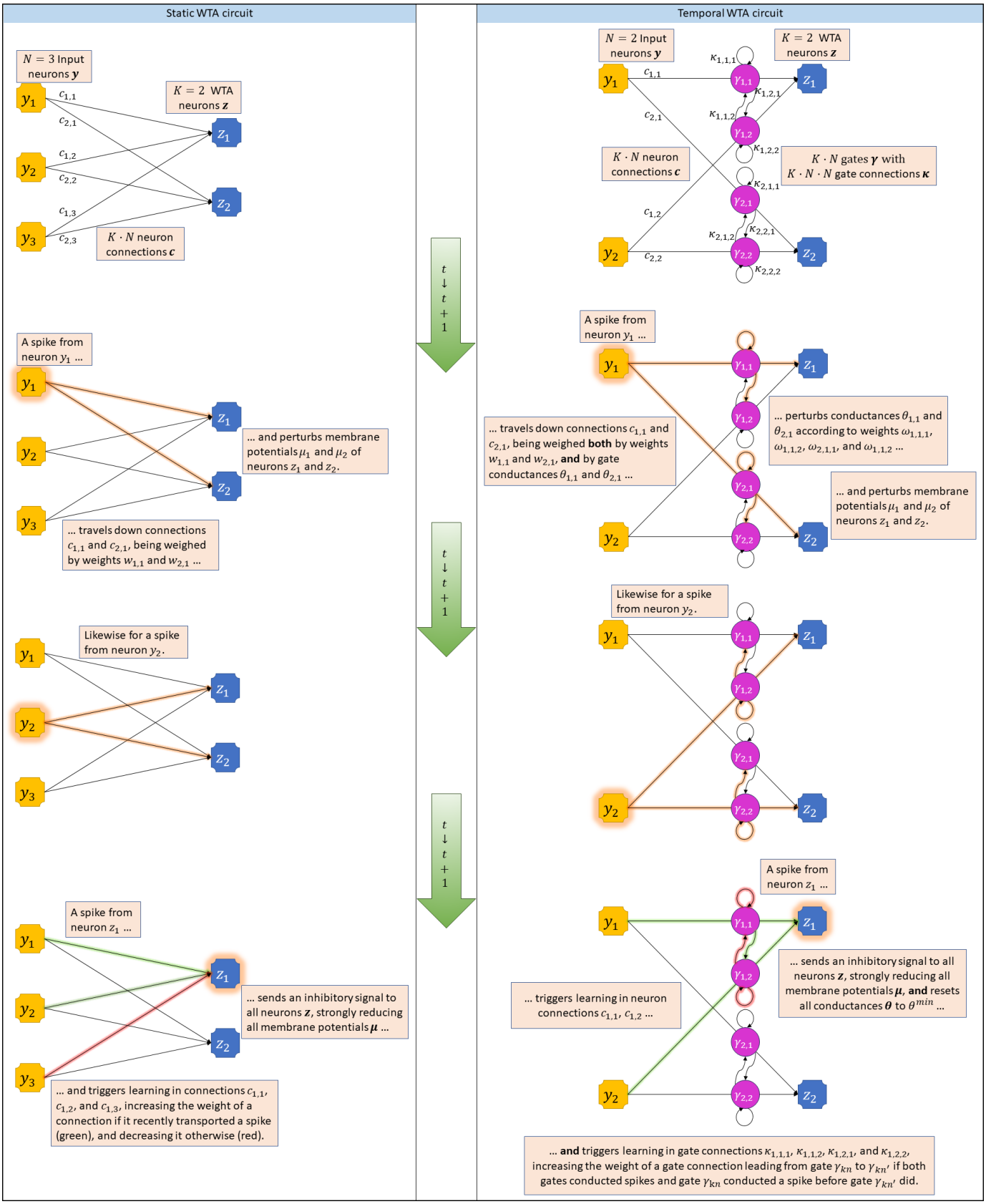


Fig. 1. A rough visualization of the dynamics of both a static WTA circuit and a temporal WTA circuit. Note that a WTA circuit has lateral connections that allow inhibitory signal $I(t)$ to be sent to all neurons $\boldsymbol{z}$ following a spike from any neuron z_k; this is not visualized here (nor modelled in detail in this work).

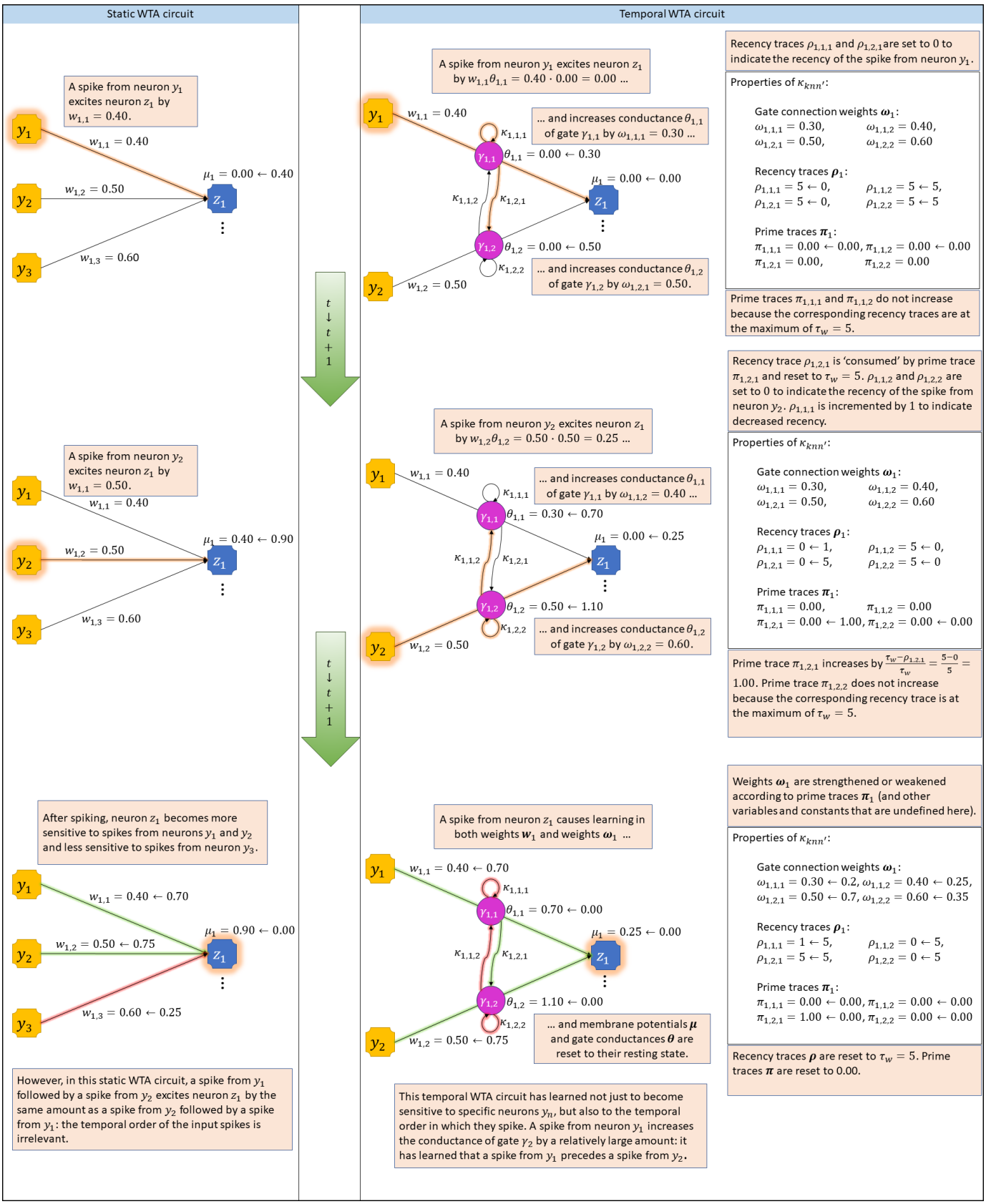


Fig. 2. A detailed visualization of the evolution of certain key variables in both a static WTA circuit and a temporal WTA circuit. The temporal WTA circuit is sensitive for a timespan of $\tau = 5$. Note that a WTA circuit has lateral connections that allow inhibitory signal $I(t)$ to be sent to all neurons $\boldsymbol{z}$ following a spike from any neuron z_k; this is not visualized here (nor modelled in detail in this work). Also note that a weight update depends on the current weight, the timing of spikes, the learning rate, and certain constants; the weight updates displayed here are just vague approximation to give a feel of the dynamics.

is encoded as a spatiotemporal spike pattern by N neurons $\boldsymbol{y}$ for the duration of T timesteps, serving as input for neurons $\boldsymbol{z}$. Whether or not a neuron y_n spikes at a given time t (i.e., whether $\phi_n(t) = 1$) depends on the stimulus and the encoding scheme.

In the experiments two types of neurons are used to implement neurons $\boldsymbol{z}$, one we refer to as stochastic neurons and the other as softmax neurons. The membrane potential of a stochastic neuron z_k is restricted to the range $[0, \mu^{max}]$. The firing probability of such a neuron increases exponentially as its membrane potential increases, according to:

$$p\big(\zeta_k(t) = 1\big) = \exp\Big(\alpha \frac{\mu_k(t) - \mu^{max}}{\mu^{max}}\Big) \tag{4}$$

Where α is a scalar constant that determines how close to μ^{max} the firing probability rapidly starts to increase[1]. Note that when μ_k is equal to μ^{max}, the firing probability of neuron z_k is equal to $e^0 = 1$. The firing probability curve is visualized for various values of μ^{max} and α in figure 3a.

The membrane potential of softmax neurons is restricted to the range $[0,\ \infty]$ and — contrary to stochastic neurons — a layer of softmax neurons spikes at pre-determined intervals. Specifically, if at a time t softmax layer $\boldsymbol{z}$ is determined to spike, a single neuron z_k is selected to produce a spike with probability:

$$p\big(\zeta_k(t) = 1\big) = \frac{e^{\mu_k(t)}}{\sum_{k'=1}^{K} e^{\mu_{k'}(t)}} \tag{5}$$

Softmax neurons provide a more predictable and controlled alternative to stochastic neurons, but unlike stochastic neurons cannot function in a local or event-based manner. This is discussed in detail in section V-D.

The membrane potentials $\boldsymbol{\mu}$ of stochastic and softmax neurons change only in response to excitatory and inhibitory spike signals. Specifically, each membrane potential $\mu_k(t)$ evolves over time according to:

$$\mu_k(t) = \mu_k(t-1) + \sum_{n=1}^{N} \phi_n(t) w_{kn}(t) - I(t) \tag{6}$$

Where variable $I(t)$ is the lateral inhibition signal elicited by spikes from neurons $\boldsymbol{z}$. In this work, the lateral inhibition signal is strong enough to ensure that the membrane potential of all neurons $\boldsymbol{z}$ is reset to zero at the next timestep, hence:

$$I(t) = \begin{cases} \infty & \text{if } \zeta(t-1) = 1 \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

Connection weights $\boldsymbol{w}$ evolve over time according to a Spike-Timing-Dependent Plasticity (STDP) rule. The weights are restricted to the range $[0,\ 1]$. Whenever a neuron z_k spikes, each of its N connection weights w_{kn} is updated according to:

$$\Delta w_{kn}(t) = f(t_{kn}^{\Delta}) \exp\big(-(w_{kn}(t) - 1)\big) - 1 \tag{8}$$

[1]As noted in table I, α and β are locally defined constants that have different meanings in different equations. Likewise $f(...)$ is a locally defined function.

Where t_{kn}^{Δ} is the time difference between the spike times of neurons z_k and y_n[2], and where $f(t_{kn}^{\Delta})$ is defined as:

$$f(t_{kn}^{\Delta}) = \frac{1}{\alpha - \beta}\left(\exp\left(-\frac{t_{kn}^{\Delta}}{\alpha}\right) - \exp\left(-\frac{t_{kn}^{\Delta}}{\beta}\right)\right) \tag{9}$$

Here α and β are constants that determine the shape of $f(t_{kn}^{\Delta})$, see figure 3b for the resulting STDP curve. Finally, each weight update is weighted by an adaptive learning rate $\eta_k(t)$ (initialized at 1.0) that diminishes each time neuron z_k spikes according to:

$$\eta_k(t) = \begin{cases} \dfrac{\eta_k(t-1)}{\eta_k(t-1)^{\hat{\eta}^{-1}} - 1} & \text{if } \zeta_k(t-1) = 1 \\ \eta_k(t-1), & \text{otherwise} \end{cases} \tag{10}$$

Where $\hat{\eta}$ is a constant that determines how quickly learning rates diminish (set to a default value of 0.60 this work). Note that each subsequent spike of neuron z_k diminishes $\eta_k(t)$ less than the one before. Finally, the resulting weight update follows:

$$w_{kn}(t) = w_{kn}(t-1) + \zeta_k(t-1)\eta_k(t-1)\Delta w_{kn}(t-1) \tag{11}$$

C. Temporal WTA circuit

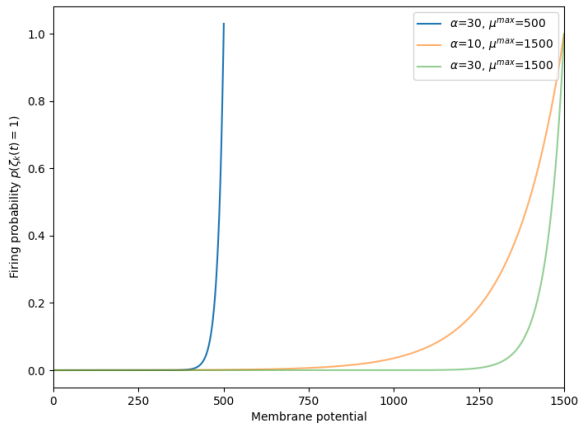
In this work we propose novel WTA circuit properties which allow WTA circuits to (learn to) distinguish not just between static input patterns, but also time-varying ones. In order to accomplish sensitivity to time-varying patterns we add what we refer to as 'gates' to the neuron connections. Each gate has a conductance that, in addition to the neuron connection weight, weighs the strength of spikes travelling through said connection. All gates of connections leading to the same neuron are interconnected, and spikes travelling through one gate influence the conductivity of other connected gates. In this fashion, the conductivity of a gate is primed by specific input spike patterns. Each gate learns to become sensitive to a separate spiking pattern via plasticity dynamics that include the maintenance of two eligibility traces. The following provides a detailed and formal description of these novel properties and dynamics.

First of all, a gate γ_{kn} is attached to each neuron connection c_{kn}. Each gate has a scalar property which we refer to as conductance $\theta_{kn}(t)$. In addition to connection weight $w_{kn}(t)$, conductance $\theta_{kn}(t)$ weighs spikes being transmitted by connection c_{kn} such that at time t a spike from neuron y_n perturbs the membrane potential of neuron z_k by an amount equal to $w_{kn}(t) \cdot \theta_{kn}(t)$:

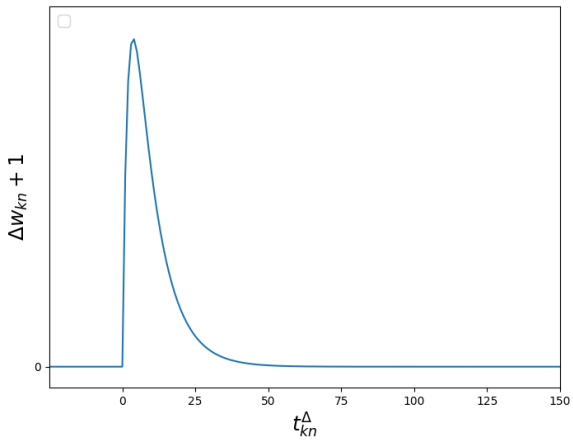
$$\mu_k(t) = \mu_k(t-1) + \sum_{n=1}^{N} \phi_n(t) w_{kn}(t) \theta_{kn}(t) - I(t) \tag{12}$$

Each gate γ_{kn} is connected to all gates $\boldsymbol{\gamma}_k$ that lead to the same neuron z_k. Hence, for each neuron z_k with N input

[2]In the case that the spike from neuron z_k was not preceded by a spike from neuron y_n, t_{kn}^{Δ} is defined to equal 0 (note that $f(0) = 0$). Further, $\Delta w_{kn}(t)$ is only computed at times t when a neuron z_k spikes. Thus, at times that $\Delta w_{kn}(t)$ is computed, a spike from neuron y_n is never more recent than a spike from neuron z_k.



(a) Firing probability $p(\zeta_k(t) = 1)$ of stochastic neurons as a function of their membrane potential, according to 4.



(b) Shape of STDP curve resulting from equation 9, where $\alpha = 2$ and $\beta = 8$.

Fig. 3

neurons $\boldsymbol{y}$, there are N neuron connections with each a gate γ_{kn} attached to it, all of which are fully connected to one another via N^2 gate connections $\boldsymbol{\kappa}_k$. This means that in a WTA circuit of K neurons $\boldsymbol{z}$ and N neurons $\boldsymbol{y}$, there are $K \cdot N^2$ gate connections $\boldsymbol{\kappa}$. The gate connections allow gates to influence the conductance of other gates. To this end, each gate connection has three variable properties.

First of all, similar to neuron connections, each gate connection $\kappa_{knn'}$ has a weight $\omega_{knn'}(t)$, which is initialized at random and restricted to the range $[0,\ 1]$. When at time t a spike travels through connection c_{kn} — and thus also through gate γ_{kn} — the conductance $\theta(t)_{kn'}$ of each of the N gates in $\boldsymbol{\gamma}_k$ is perturbed by an amount equal to $\omega_{knn'}(t)$. Specifically, conductance $\theta(t)_{kn}$ evolves over time according to:

$$\theta_{kn}(t) = \theta_{kn}(t-1) + \sum_{n'=1}^{N} \phi_{n'}(t-1)\omega_{knn'}(t-1) \qquad (13)$$

In our experiments, the conductance of each gate is initialized at 0 at the onset of each stimulus, and is restricted to the range $[0,\ \infty]$.

Like in a static WTA circuit, the intention is for each neuron z_k to grow sensitive to a specific pattern of sensory spikes. Unlike a static WTA circuit, a temporal WTA circuit is intended to extract such information from the temporal order in which sensory spikes arrive. The evolution of the conductances is meant to facilitate this. For example, if neuron z_k responds often to a pattern where a spike from neuron $y_{n'}$ precedes a spike from neuron y_n, then weight $\omega_{knn'}(t)$ should grow to be large, such that a spike from $y_{n'}$ strongly increases conductance $\theta_{kn}(t)$, and thereby the excitation delivered to z_k by a subsequent spike from y_n.

In order to realize such an evolution of weights $\boldsymbol{\omega}(t)$, each gate connection maintains two eligibility traces that serve as a short-term memory of local events. First, each gate connection $\kappa_{knn'}$ maintains a scalar 'recency trace' $\rho_{knn'}(t)$. Each recency trace is initialized at τ at the onset of a new stimulus and restricted to the range $[0,\ \tau]$, where constant scalar τ represents the timescale at which the gates operate. Specifically, if a circuit produces a spike every 50 timesteps, then τ is set to 50, and the circuit is able to grow sensitive to patterns spanning up to 50 timesteps. Recency trace $\rho_{knn'}(t)$ serves as memory of how many timesteps ago (up to τ timesteps) neuron $y_{n'}$ produced a spike.

Second, each gate connection $\kappa_{knn'}$ maintains a scalar 'prime trace' $\pi_{knn'}(t)$. Each prime trace is initialized at 0 at the onset of each new stimulus and is restricted to the range $[0,\ \infty]$. When a neuron y_n spikes at time t, each prime trace in $\boldsymbol{\pi}_{kn}(t)$ is increased proportionally to the recency of spikes from each neuron in $\boldsymbol{y}$, as remembered by recency traces $\boldsymbol{\rho}_{kn}(t)$. Each prime trace is a measure of how closely and how consistently a spike from neuron y_n was preceded by a spike from neuron $y_{n'}$. Whenever a neuron z_k spikes this information is used to determine the strength of gate connection weight updates.

The exact dynamics according to which gate connection properties $\omega_{knn'}(t)$, $\rho_{knn'}(t)$, and $\pi_{knn'}(t)$ evolve are as follows. At each point in time, these properties are updated sequentially in the order that they are here mentioned. First, each recency trace $\rho_{knn'}$ is incremented by one:

$$\rho_{knn'}(t) = \rho_{knn'}(t-1) + 1 \qquad (14)$$

This represents the fact that, with the passage of time, spikes have become less recent.

Second, prime trace $\pi_{knn'}$ evolves according to:

$$\pi_{knn'}(t) = \pi_{knn'}(t-1) + \phi_n(t)\frac{\tau - \rho_{knn'}(t)}{\tau} \qquad (15)$$

Which entails that, given a spike from neuron y_n at time t, prime trace $\pi_{knn'}(t)$ is increased by an amount proportional to recency trace $\rho_{knn'}(t)$. Note that this increase progresses linearly from 1 to 0 as $\rho_{knn'}(t)$ progresses from 0 to τ.

Third, following the spike from neuron y_n, the recency traces used to update $\boldsymbol{\rho}_{kn}(t)$ are reset to τ (except where

$n' = n$):

$$\forall n' \neq n : \rho_{knn'}(t) = \begin{cases} \tau & \text{if } \phi_n(t) = 1 \\ \rho_{knn'}(t), & \text{otherwise} \end{cases} \quad (16)$$

While the recency traces indicating the recency of a spike from neuron y_n are set to zero (indicating maximum recency):

$$\rho_{kn'n}(t) = \begin{cases} 0 & \text{if } \phi_n(t) = 1 \\ \rho_{kn'n}(t), & \text{otherwise} \end{cases} \quad (17)$$

Fourth, a spike from a neuron z_k elicits learning in gate connection weights $\boldsymbol{\omega}_k$ according to:

$$\Delta\omega_{knn'}(t) = \pi_{knn'}(t) \exp\big(-(\omega_{knn'}(t) - 1)\big) - 1 \quad (18)$$

$$\omega_{knn'}(t) = \omega_{knn'}(t-1) + \zeta_k(t-1)\eta_k(t-1)\Delta\omega_{knn'}(t-1) \quad (19)$$

Note the similarity between equations 18 and 19, and equations 8 and 11: $\omega_{knn'}(t)$ takes the place of $w_{kn}(t)$ and $\pi_{knn'}(t)$ takes the place of $f(t_{kn}^{\Delta})$. A weight update increases as the corresponding prime trace increases, and decreases as the to-be-updated weight grows larger. Like the updates of neuron connections weights, the update of gate connection weights are restrained by adaptive learning rate $\eta_k(t)$.

Finally, following the spike from neuron z_k, all recency traces $\boldsymbol{\rho}$ and all prime traces $\boldsymbol{\pi}$ are reset to their initial states:

$$\rho_{knn'}(t) = \rho_{knn'}(t) + \zeta(t)\tau \quad (20)$$

$$\pi_{knn'}(t) = \zeta(t)\pi_{knn'}(t) \quad (21)$$

Thus, unless the input pattern repeats, whatever has been learned by neuron z_k will not again be learned following subsequent spikes from neurons $\boldsymbol{z}$.

The expectation is that the above dynamics cause each neuron z_k in a temporal WTA circuit to become distinctly sensitive to a unique temporal order of input spikes from neurons y_n. In order for this to be, it is expected that at least one condition must hold: the circuit must operate at a timescale that matches the duration of the input patterns. If patterns repeat before the circuit produces a spike, then learned information about the temporal order of input spikes will be muddled. Furthermore, given that weights $\boldsymbol{\omega}$ only grow when the corresponding input neurons spiked within some time of one another, the expectation is that sparse input spikes will cause weights $\boldsymbol{\omega}$ to evolve to become sparse themselves. These expectations are addressed and tested in the experiments.

IV. Experiments

Several experiments have been performed in order to gauge the behaviour and performance of temporal WTA networks[3]. The experiments vary in the data used, the manner in which the data is encoded, and in the network architectures that are employed. Each architecture consists of a single layer of sensory neurons $\boldsymbol{y}$ which encode the data over time in the form of spike trains. Following this input layer are one or more layers that each consist of one or more WTA circuits which receive input from the preceding layer. The final layer is always a single WTA circuit that consists of softmax neurons. The final layer produces a single spike the end of each stimulus, which is considered to be the network's classification of the preceding stimulus.

Each experiment consists of multiple runs using different random number generator seeds, the performance being averaged over all these runs. Each run consists of a training phase, a mapping phase, and a testing phase. Each phase entails one or more cycles over the selected data. First, during the training phase, stimuli are presented to the network in the form of spike trains emitted by sensory neurons $\boldsymbol{y}$. This elicits the propagation of spiking activity throughout the rest of the WTA network, and plasticity dynamics cause network weights to evolve and become sensitive to distinct spiking patterns. Weights are only allowed to evolve during the training phase, and are frozen during the subsequent mapping and testing phases.

Second, in the mapping phase, it is determined which neuron is associated with which class. Given that the WTA network learns in an entirely unsupervised fashion, it has itself no notion of stimulus classes. Thus, in order to associate neurons with a specific class, a mapping from each neuron to a specific class must be decided externally. This is done by observing the response of each neuron to stimuli during the mapping phase. At the end of this phase, the class to which a neuron responded the most is considered to be that neuron's class, and a spike from that neuron is considered to be a classification of that class.

Third and finally, in the testing phase, the accuracy of the network output is determined using the established neuron-to-class mapping, indicating how well each output neuron has become sensitive to a specific class. When data used in the mapping and testing phase is the same, it is possible that the accuracy is increased due to the selection of an optimal mapping based on test data. However, as the experiments show, this effect is negated by using sufficient amounts of data and by keeping the mapping and testing phases separate. In addition to accuracy, the experiments determine the sparsity of gate connection weights following training.

A. Data

Three datasets are used throughout the experiments. In the following these datasets are described, as well as the manner in which these are encoded as spike trains.

1) Toy data: The first dataset is a self-constructed toy dataset consisting of four black-and-white images of 10×10 pixels. The four images are displayed in figure 4. Throughout the experiments these images are encoded as spike trains in two fashions.

First of all, the dataset is converted to spike trains via what we refer to as a static encoding. This encoding is similar to the encoding method used to encode the MNIST dataset in the works of [41]–[43][4]. Using the static encoding, each pixel is

[3]The code used to perform the experiments can be found on https://github.com/Grottoh/Temporal-WTA-Network.

[4]The encoding is similar, but not the same. In previous works each black-and-white pixel is encoded by two neurons, one being active if the corresponding pixel value is black, the other being active if the corresponding pixel value is white. In this work a single neuron is used to encode each pixel, and is only active when its value is white.

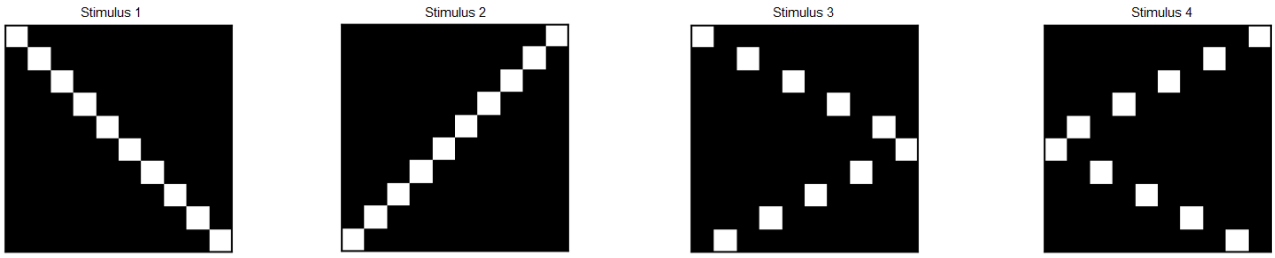
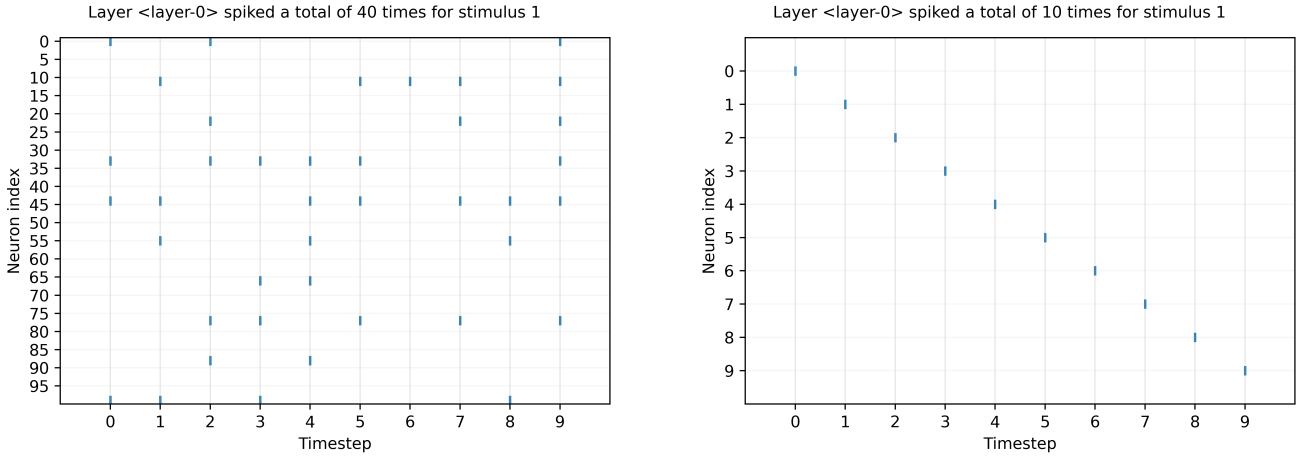


Fig. 4. The toy data: four 10×10 black-and-white images.



(a) Static encoding of the first image of the toy data.

(b) Temporal encoding of the first image of the toy data.

Fig. 5. Toy data.

encoded by a single input neuron. If the pixel is white, then the corresponding input neuron is considered to be active. Active input neurons produce Poisson spike trains over some predetermined duration T (here set to 10), where at each timestep they have a constant probability to spike. Thus, each image is encoded by 10×10 sensory neurons over a period of 10 timesteps. The fact that for a given stimulus the same neurons are active at all times, and the fact that the spiking probability of active neurons is constant, is why we refer to this as a static encoding scheme. Figure 5a shows an example of spike trains resulting from a static encoding of toy data.

Secondly, the toy data is encoded via what we refer to as a temporal encoding, where one dimension of the data is folded out over time. Specifically, at timestep t, neuron y_n produces a spike if and only if the pixel in row n and column t of the image is white. Thus, the encoding requires 10 neurons (one for each row) and 10 timesteps (one for each column) to encode an image. Note that, unlike in the static encoding, the spiking probability of each neuron in $\boldsymbol{y}$ changes over time, as such the temporal order in which neurons $\boldsymbol{y}$ spike becomes relevant. Figure 5b shows an example of spike trains resulting from a temporal encoding of toy data.

2) Concatenated toy data: The second dataset is a concatenation of the the toy data. Specifically, four new images are created by forming four pairs of the original toy images, as visualized in figure 6. Encoding this concatenated toy data via the temporal encoding scheme requires 20 rather than 10 timesteps, but otherwise proceeds in the same fashion.

3) TIDIGITS: The third dataset is the TIDIGITS dataset [44]. The TIDIGITS dataset consists of speech samples from 326 speakers each speaking 22 single-digit utterances[5], yielding a total of $7,172$ separate single-digit utterances. Half of the data ($3,586$ utterances) is assigned to be training data, the other half to be test data, with no overlap in speakers between these two partitions. Several classes are assigned to each utterance:

- Digit (11 equally distributed classes): zero, oh, one, two, three, four, five, six, seven, eight, nine
- Speaker ID (326 classes): each of the 326 speakers has a unique ID (e.g., 'aa' or 'tc')
- Demographic (4 classes): boy, girl, man, woman

Before the TIDIGITS dataset is encoded as spikes, the raw audio files of which it consists are converted to Mel-Frequency Cepstral Coefficients (MFCCs). Specifically, each utterance is represented by 13 MFCCs and 13 MFCC deltas (the latter of which represent the change between MFCC time-frames). Thus a single utterance x is represented by $26 \cdot T_x$ scalar values, where T_x is the duration of utterance x in terms of MFCC time-frames. The MFCC representation of spoken digit 'one' is displayed in figure 7a.

In order to encode the MFCCs (and MFCC deltas) as spikes, each coefficient is divided into B bins. The range of values

[5]The dataset also contains digit-sequence utterances, but these are not used in this work.

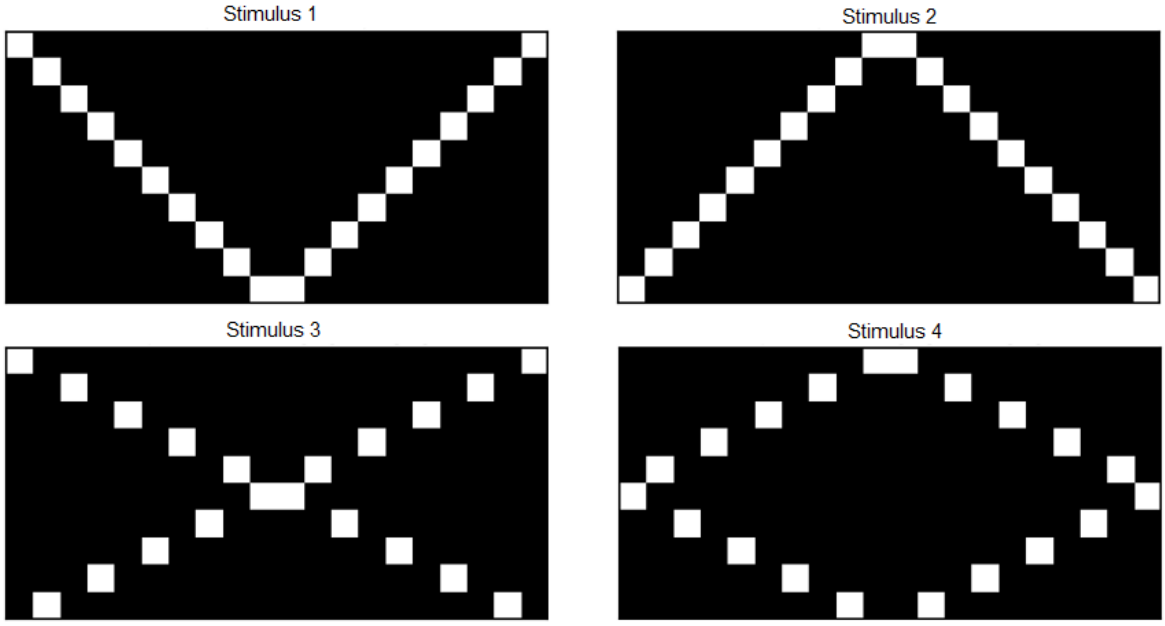
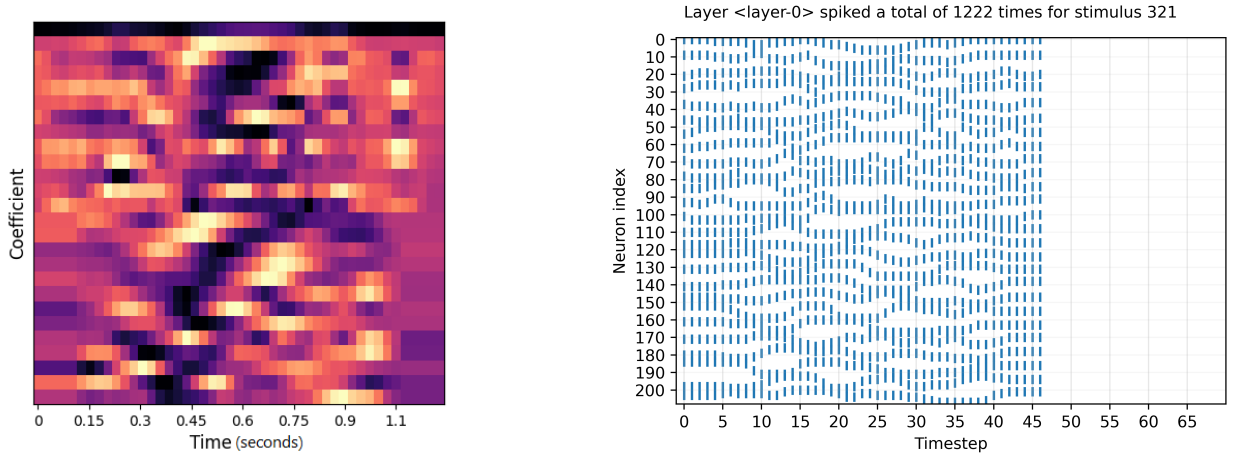


Fig. 6. Concatenated toy data: four 10×20 (rows $\times$ columns) black-and-white images.



(a) Visualization of MFCCs and their deltas of digit 'one' being spoken.

(b) Spike encoding of the coefficients displayed in figure 7a ($B = 8$).

Fig. 7. TIDIGITS data.

covered by each bin is determined by the distribution of its corresponding coefficient over the entire dataset, such that a bin covering a range of frequently appearing values will be smaller than a bin covering a range of rare values. Specifically, each bin covers $\frac{1}{B}\%$ of the coefficient values over the entire dataset. A neuron is assigned to each bin, yielding $26 \cdot B$ neurons $\boldsymbol{y}$. Each MFCC time-frame is encoded in a single timestep, such that each utterance is encoded in T_x timesteps. At each timestep, a neuron y_n spikes if the bin it encodes covers the value of the coefficient at the corresponding time-frame, yielding exactly 26 spikes at every point in time. Figure 7b shows how the spoken digit 'one' is encoded as spikes. Additional discussion on this encoding of the TIDIGITS data follows in section V-C.

B. Network architectures

The experiments test the performance of three different network architectures. The first two network architectures are the static WTA circuit described in section III-B and the temporal WTA circuit described in III-C. At the end of section III-C the assertion was made that temporal WTA circuits must operate at a timescale that matches the duration of input patterns. For both the concatenated toy data and the TIDIGITS data, it is expected that the corresponding spike patterns cannot be distinguished as a whole by temporal WTA circuits. Instead, it is expected that these spike patterns must be processed on multiple timescales: a smaller timescale where smaller temporal patterns are extracted, and a larger timescale where the larger temporal pattern is derived from the smaller ones. Our solution to this problem is to employ a two-layer temporal

WTA network.

Our two-layer temporal WTA network architecture consists first of all of a WTA layer $\mathcal{L}_1$ that itself consists of $\mathcal{N}_1$ temporal WTA circuits. Each of the circuits in $\mathcal{L}_1$ consists of K_1 neurons that receive input from all sensory neurons $\boldsymbol{y}$ via gated neuron connections. Secondly, it consists of a WTA layer $\mathcal{L}_2$ that is a single temporal WTA circuit. The single circuit of $\mathcal{L}_2$ consists of K_2 softmax neurons which receive input from all neurons in $\mathcal{L}_1$ via gated neuron connections. The idea is that layer $\mathcal{L}_1$ is able to distinguish between relatively short spike patterns produced by sensory neurons $\boldsymbol{y}$, and that $\mathcal{L}_2$, operating at a larger timescale, in turn distinguishes between the spike patterns produced by $\mathcal{L}_1$ (akin to how one might divide speech into syllables, and then words). An overview and description of this architecture's hyperparameters is provided per experiment where it is employed.

C. Experiment 1

1) Setup: In experiment 1 the capabilities of a static WTA circuit is compared to that of a temporal WTA circuit. The performance of these two types of networks is assessed both on the static and the temporal encoding of the toy data (described in section IV-A1). The purpose of this experiment is to provide experimental evidence for our assertion that static WTA circuits are incapable of extracting temporal relations from stimuli, whereas our proposed temporal WTA circuit is.

Experiment 1 is divided into two experiments, which are each divided into more sub-experiments. Experiment 1.1 assesses the ability of a static WTA circuit with $K = 4$ softmax neurons to learn to distinguish between the four unique toy stimuli. Experiment 1.2 assesses the ability of a temporal WTA circuit with $K = 4$ softmax neurons to learn to distinguish between the four unique toy stimuli.

Each sub-experiment is run 10 times with different random number generator seeds. By default each run consists of 1 train cycle, 10 map cycles, and 10 test cycles. Note that the same data is used for training, mapping, and testing. The results of experiment 1 are displayed in table III. Examples of spike patterns and learned weights are shown in figures 8 and 9.

2) Interpretation: Experiment 1.1 shows that the static WTA circuit is capable of perfectly distinguishing between the toy stimuli when these are encoded via the static encoding scheme. It further shows that when these same stimuli are encoded via the temporal encoding scheme, the performance of the circuit essentially reduces to that of random guessing (25%). This is in line with our assertion that static WTA circuits are unsuitable for distinguishing between time-varying stimuli.

Experiment 1.2 shows that the temporal WTA circuit is capable of perfectly distinguishing between the toy stimuli, both when using the static encoding scheme and when using the temporal encoding scheme. This is evidence for our assertion that the temporal WTA circuit is capable of distinguishing between time-varying (as well as static) stimuli. Experiment 1.1c further shows that, at least for this simple task, neuron connection weights $\boldsymbol{w}$ can be omitted without hurting performance, and thus that gate connection weights $\boldsymbol{\omega}$ are sufficient for distinguishing between simple time-varying spiking patterns.

D. Experiment 2

1) Setup: At the end of section III-C the assertion was made that temporal WTA circuits cannot handle repeating patterns. When a larger pattern consists of smaller patterns that each involve spikes from the same neurons in $\boldsymbol{y}$, then these smaller patterns must be distinguished in order for a temporal WTA circuit to recognize the larger pattern. The expectation is that this can be achieved by combining multiple circuits into a network, as is done in the two-layer WTA network introduced in section IV-B.

The concatenated toy data is used to test the above assertions. Note that, when representing this data using the temporal encoding, each sensory neuron will fire exactly twice. Note especially that at least one spike from each sensory neuron precedes a spike from each other sensory neuron. A temporal WTA circuit meant to distinguish between these stimuli is expected to fail, because for each stimulus it will learn (correctly) that each neuron produces a spike before every other neuron. In the two-layer network, however, layer $\mathcal{L}_1$ (operating at timescale $\tau = 10$) can distinguish between the original non-repeating toy patterns of 10 timesteps, whilst layer $\mathcal{L}_2$ (operating at timescale $\tau = 20$) uses input from $\mathcal{L}_1$ to determine the concatenated pattern of 20 timesteps.

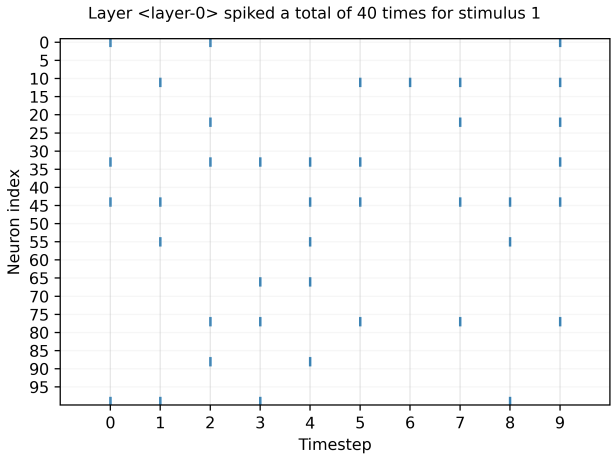
Experiment 2 is divided into three experiments, some of which are divided into more sub-experiments. Experiment 2.1 assesses the performance of a temporal WTA circuit. Experiment 2.2 assesses the performance of a two-layer temporal WTA network with softmax neurons in layer $\mathcal{L}_1$. Experiment 2.3 assesses the performance of a two-layer temporal WTA network with stochastic neurons in layer $\mathcal{L}_1$. The final layer of each network is set to consists of 4 softmax neurons, and neuron connection weights are fixed to 1.0.

Each sub-experiment is run 10 times with different random number generator seeds. By default each run consists of 10 train cycles, 10 map cycles, and 10 test cycles. Note that the same data is used for training, mapping, and testing. The results of experiment 2 are displayed in table IV. Examples of spike patterns and learned weights are shown in figures 10, 11, 12 and 13.

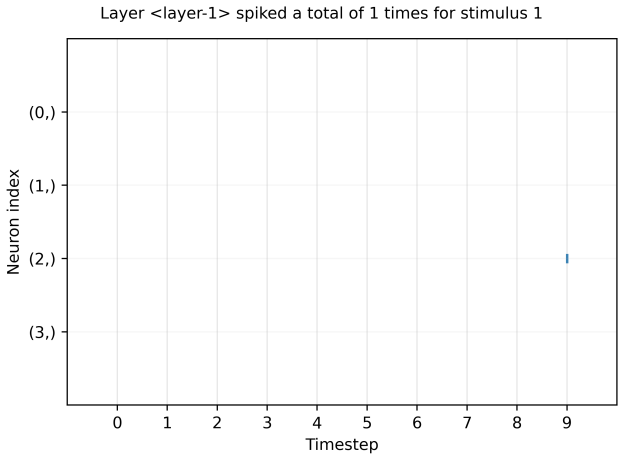
2) Interpretation: The chance accuracy achieved in experiment 2.1 provides evidence for our assertion that temporal WTA circuits are unsuitable for distinguishing between repeating patterns. Figure 10 shows that each neuron z_k learns that a spike from each neuron y_n, to some extent, precedes a spike from each other neuron $y_{n'}$. While this is in fact true, it prohibits the circuit from distinguishing between the larger patterns.

Experiment 2.2 shows how this problem is solved by employing the two-layer network. Layer $\mathcal{L}_1$, operating at a timescale of $\tau = 10$, recognizes the original toy patterns. Layer $\mathcal{L}_2$, operating at a timescale of $\tau = 20$, is then able to distinguish between the concatenated patterns based on the spiking activity of $\mathcal{L}_1$.

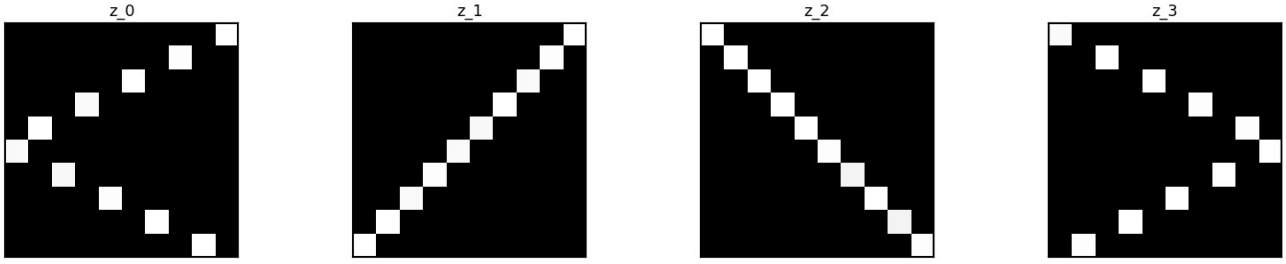
While experiment 2.2 uses softmax neurons in layer $\mathcal{L}_1$, experiment 2.3 uses stochastic neurons. As discussed in more



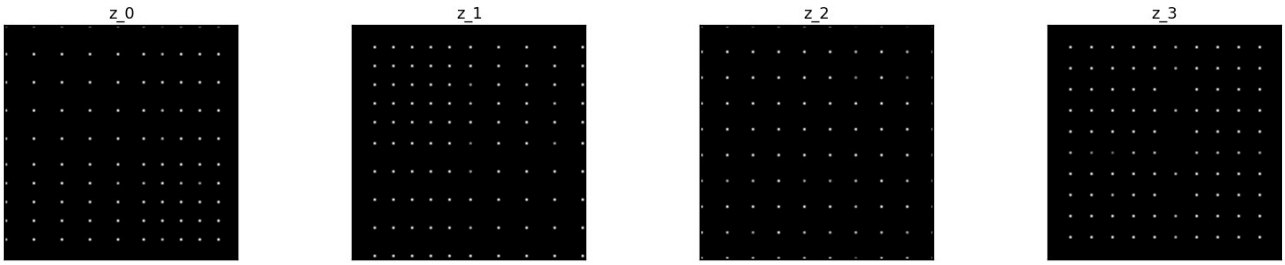
(a) Spikes resulting from a static encoding of the first image of the toy data.



(b) The WTA circuit produces one spike at the end of the stimulus.



(c) Visualization of the **neuron** connection weights following the train phase of experiment 1.1a. Each neuron z_k has a weighted connection to each input pixel (represented by a sensory neuron y_n). Note how the neuron connection weights come to exactly resemble to toy data (see 4)



(d) Visualization of the **gate** connection weights following the train phase of experiment 1.2a. It shows — for the stimulus that a neuron has grown sensitive to — that a spike from each active neuron is considered to predict a spike from each other active neuron

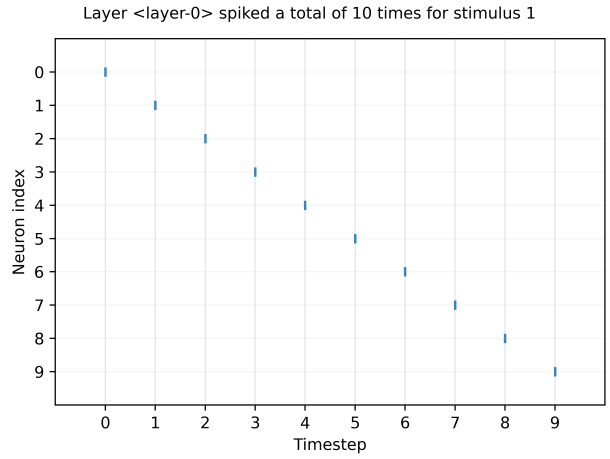
Fig. 8. Visualizations corresponding to experiments 1.1a and 1.2a.

detail in section V-D, softmax neurons violate various neuromorphic principles. Stochastic neurons provide a biologically realistic alternative, but operate in a less controlled and less predictable manner. Figure 12 shows how in experiment 2.3a the stochastic neurons of $\mathcal{L}_1$ produce spikes at intervals that do not match the duration of the original toy data. As a result, each neuron in $\mathcal{L}_1$ learns to distinguish between sub-optimal patterns (given our performance measure). $\mathcal{L}_1$ thus provides flawed information to layer $\mathcal{L}_2$, which achieves a sub-optimal accuracy of 72.50%.

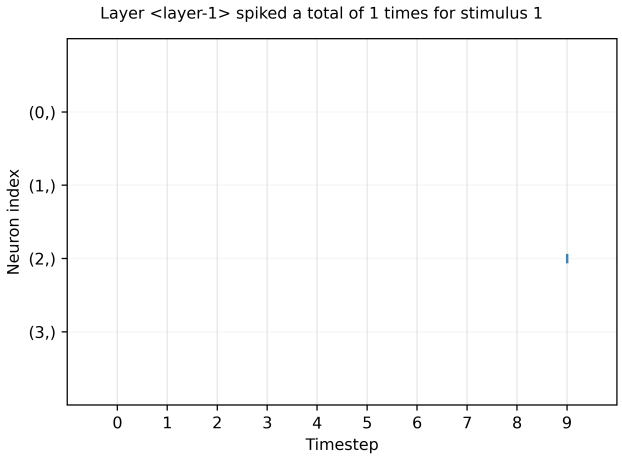
Because the stochastic neurons do not perfectly capture the duration of the largest non-repeating patterns (10 timesteps), more neurons are required in $\mathcal{L}_1$ to encode a larger number of sub-optimal patterns. Experiment 2.3b shows how this improves the accuracy to 90.25%. Furthermore, experiment 2.3c shows how additionally increasing the number of circuits in layer $\mathcal{L}_1$ allows the network to achieve perfect accuracy on all runs. This shows how the randomness inherent to stochastic neurons can be averaged out by the increasing the number of neurons and circuits.

E. Experiment 3

1) Setup: In experiment 3 the behaviour and performance of the two-layer temporal WTA network architecture is assessed on the TIDIGITS dataset. Specifically, each run uses data from a single speaker. Each speaker speaks two utterances for each of the 11 single-digit classes, yielding a total 22 unique utterances per speaker. By default, layer $\mathcal{L}_2$ consists of $K_2 = 22$ softmax neurons. The purpose of this experiment is to determine whether the two-layer network is capable of remembering not just the simple toy spiking patterns, but also

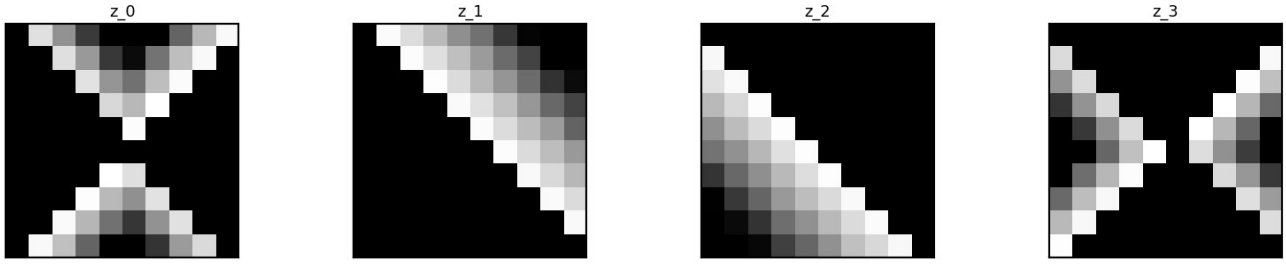


(a) Spikes resulting from a temporal encoding of the first image of the toy data.



(b) The WTA circuit produces one spike at the end of the stimulus.

(c) Visualization of the **neuron** connection weights following the train phase of experiment 1.1c.



(d) Visualization of the **gate** connection weights following the train phase of experiment 1.2b. Neuron z_2 has grown sensitive to stimulus 1, having learned that a spike from neuron y_0 is preceded by nothing, that a spike from neuron y_1 is preceded by a spike from neuron y_0 (hence why pixel $(1, 0)$ is white), and so forth.

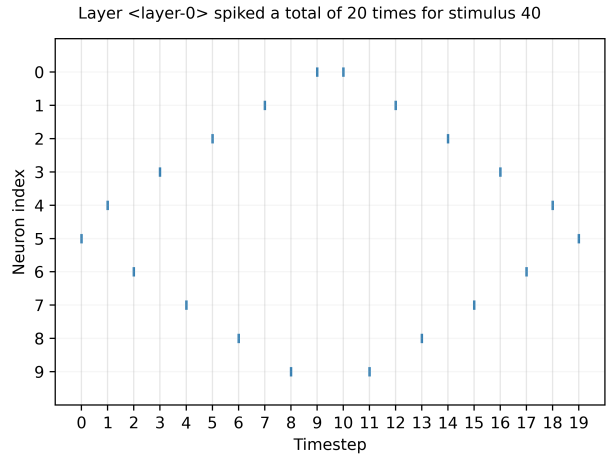
Fig. 9. Visualizations corresponding to experiments 1.1c and 1.2b.

more complex spiking patterns extracted from speech. This is the case if, after training, each of the 22 output neurons responds only to a single unique utterance.

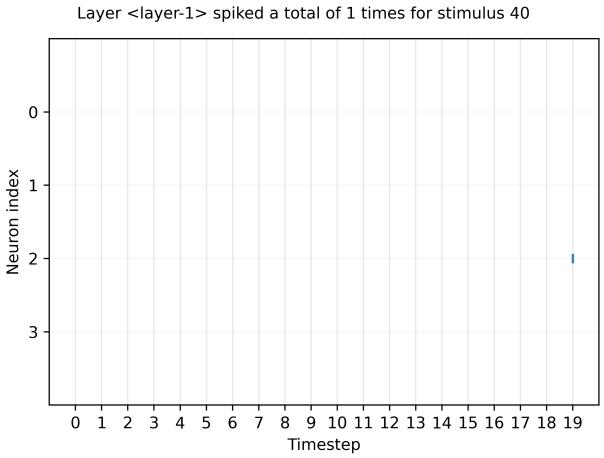
Experiment 3 is divided into three experiments, which are each divided into more sub-experiments. Experiment 3.1 assesses the impact of the number of train cycles. Experiment 3.2 assesses the impact of the number of circuits $\mathcal{N}_1$ in layer $\mathcal{L}_1$. Finally, experiment 3.3 assesses the impact of the number of neurons K_1 and K_2 in layers $\mathcal{L}_1$ and $\mathcal{L}_2$.

Each sub-experiment is run 5 times with different seeds and is repeated for 8 different speakers (2 boys, 2 girls, 2 men, 2 women). By default, each run consists of 10 train cycles, 10 map cycles, and 10 test cycles. Note that the same data is used for training, mapping, and testing. Additional default hyperparameters are displayed in table II. The results of experiment 3 are displayed in table V. Examples of spike patterns and learned weights are shown in figure 14.

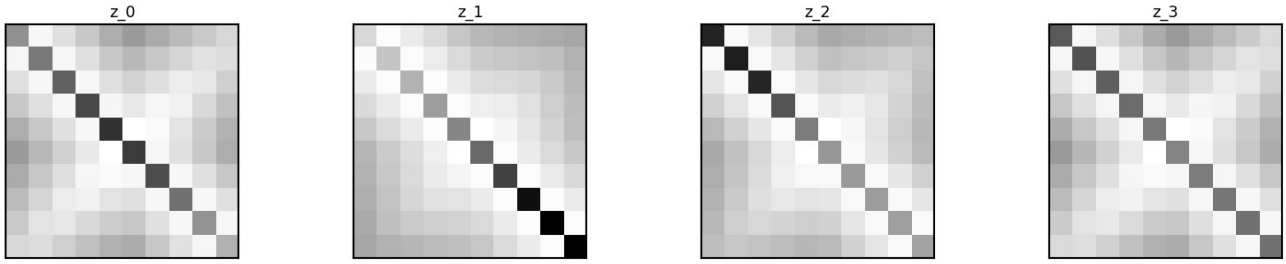
2) Interpretation: Experiment 3.1 shows that the temporal WTA network can learn to remember not just simple toy spiking patterns, but also more complex spiking patterns that encode speech. The results show that, on nearly all runs, the network establishes a one-to-one mapping from neuron to utterance within 5 train cycles. Experiment 3.2 shows that this result is achieved more consistently when including a higher number of circuits in layer $\mathcal{L}_1$ of the network. Experiment 3.3a shows that the performance drops when having too few neurons in the circuits of $\mathcal{L}_1$. Finally, experiments 3.3c and 3.3d show that increasing the number of neurons in layer $\mathcal{L}_2$ does not hurt performance, though it requires more map cycles to establish an optimal mapping.



(a) Spikes resulting from temporal encoding of first image of repeated toy data.



(b) The WTA circuit produces one spike at the end of the stimulus.



(c) Visualization of the **gate** connection weights following the train phase of experiment 2.1. Each neuron has learned that a spike from each neuron predicts a spike from each other neuron (which taken over the course of the entire stimulus, is true for all stimuli).

Fig. 10. Visualizations corresponding to experiment 2.1.

F. Experiment 4

1) Setup: The setup of experiment 4 is largely identical to that of experiment 3. Like experiment 3, each run in experiment 4 concerns 22 utterances from a single TIDIGITS speaker. Further, like in experiment 3, the network under consideration is the two-layer temporal WTA network. The main difference is that the number of neurons in layer $\mathcal{L}_2$ is here set to $K_2 = 11$, making a one-to-one or one-to-many mapping from utterance to neuron impossible. Thus, instead of remembering each unique utterance, experiment 4 assesses whether each neuron becomes distinctly sensitive to a specific digit (recall that the 22 utterances concern 2x11 single-digit utterances).

Experiment 4 is divided into three experiments, which are each divided into more sub-experiments. Experiment 4.1 assesses the impact of the number of train cycles. Experiment 4.2 assesses the impact of the number of circuits $\mathcal{N}_1$ in layer $\mathcal{L}_1$. Finally, experiment 4.3 assesses the impact of the number of neurons K_1 in layer $\mathcal{L}_1$.

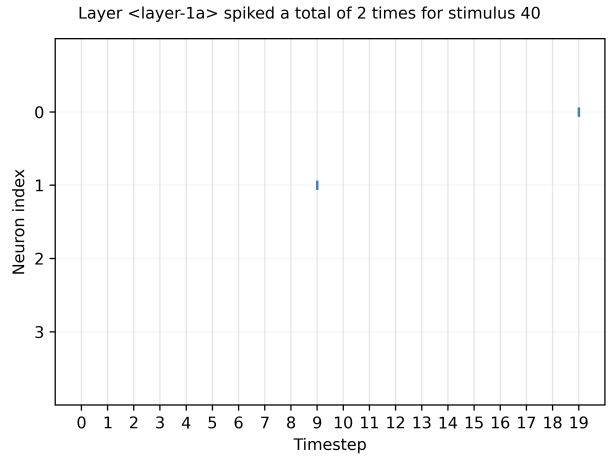
Each sub-experiment is run 5 times with different seeds and is repeated for 8 different speakers (2 boys, 2 girls, 2 men, 2 women). By default, each run consists of 30 train cycles, 10 map cycles, and 10 test cycles. Note that the same data is used for training, mapping, and testing. Additional default hyperparameters are displayed in table II. The results of experiment 4 are displayed in table VI. Examples of spike patterns and learned weights are shown in figure 14.

2) Interpretation: If the two-layer network simply 'remembers' utterances, like experiment 3 shows it can, then with $K_2 = 11$ output neurons it can remember half of the 22 utterances. If the learned information has no bearing at all on the other 11 utterances, then one would expect chance accuracy of $\frac{1}{11} \approx 9.09\%$ on these utterances, resulting in an expected accuracy of 54.45% over all 22 utterances. The experiments achieve up to 88.86% accuracy (see experiment e4.1d). This shows that, beyond remembering, the two-layer network is capable of extrapolating to digit classes.

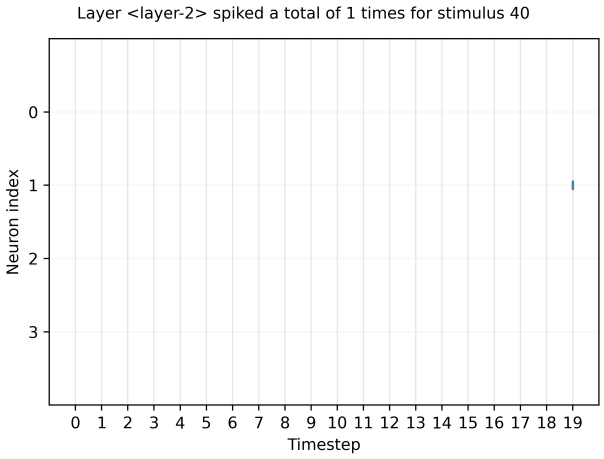
Unlike experiment 3.2, experiment 4.2 shows no clear difference in performance when including a higher number of circuits in layer $\mathcal{L}_1$. However, like experiment 3.3a, experiment 4.3 shows that the performance drops when having too few neurons in the circuits of $\mathcal{L}_1$.

G. Experiment 5

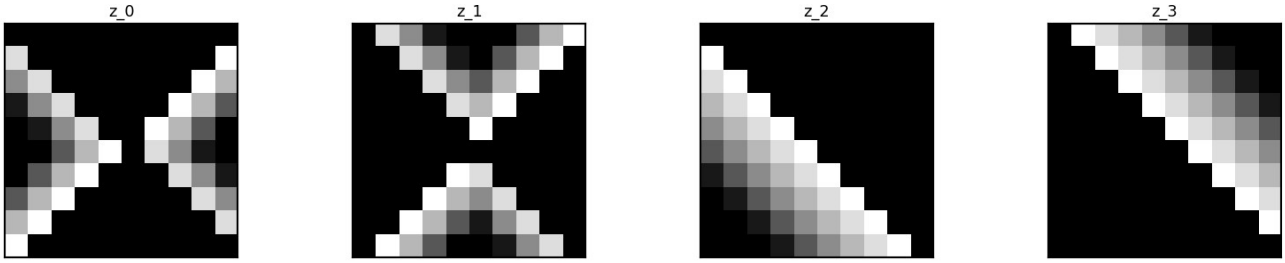
1) Setup: In experiment 5 the behaviour and performance of the two-layer temporal WTA network architecture is assessed on the entire (single-digit) TIDIGITS dataset. By default, layer $\mathcal{L}_2$ consists of $K_2 = 100$ softmax neurons. The purpose of this experiment is to assess how well each neuron in output layer $\mathcal{L}_2$ becomes distinctly sensitive to a specific digit class. Using the entire TIDIGITS dataset introduces a lot more variation in the data, making the task more difficult than it is in experiment 4. Furthermore, due to the separation of train and test data, remembering specific utterances during testing is not possible.



(a) Spikes produced by the first (and only) circuit of layer $\mathcal{L}_1$.



(b) Output spike produced by the single circuit of layer $\mathcal{L}_2$.



(c) Visualization of the **gate** connection weights of the first (and only) circuit of layer $\mathcal{L}_1$ following the train phase of experiment 2.2. It shows how circuits in layer $\mathcal{L}_1$ learn the patterns of the original (unrepeated) toy data.

Fig. 11. Visualizations corresponding to experiment 2.2. The figure shows how a layer operating at a smaller timescale can extract (and spike-encode) features, which can be used by a subsequent layer operating at a larger timescale to perform classification. Spikes of the sensory layer are displayed in figure 10a.

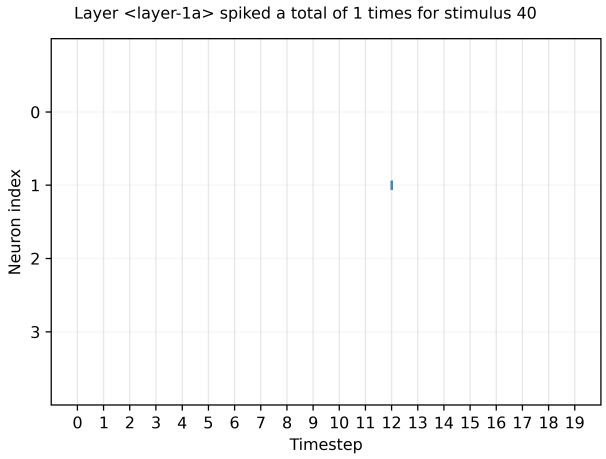
In order to give an impression of how the two-layer network behaves under different circumstances and settings, experiment 5 assesses performance under many different hyperparameter settings.

Experiment 5 is divided into seven experiments, which are each divided into more sub-experiments. Experiment 5.1 assesses the impact of the number of train cycles. Experiment 5.2 assesses the impact of the number of circuits $\mathcal{N}_1$ in layer $\mathcal{L}_1$. Experiment 5.3 assesses the impact of the number of neurons K_1 and K_2 in layers $\mathcal{L}_1$ and $\mathcal{L}_2$. Experiment 5.4 assesses the impact of the value of learning rate decay $\hat{\eta}$. Experiment 5.5 assesses impact of the maximum allowed number of simultaneous spikes of circuits in $\mathcal{L}_1$. Experiment 5.6 assesses the impact of certain variations in the data encoding. Experiment 5.7 assesses the impact of miscellaneous variations: (a) uses shuffled data, (b) allows learning with respect to neuron connection weights, (c) combines several optimal parameters from earlier experiments, and (d) assesses performance with respect to demographic classes.

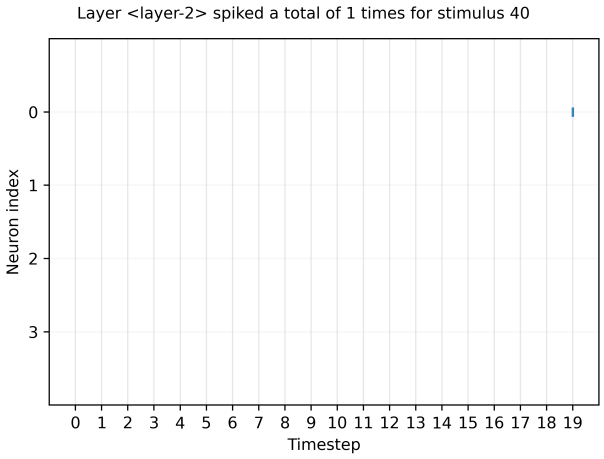
Each sub-experiment is run 3 times with different seeds for the entire (single-digit) TIDIGITS dataset, which consists of $3,586$ train utterances and $3,586$ test utterances. By default, each run consists of 5 train cycles, 1 map cycle, and 1 test cycle. Training and mapping is done on the train data, whereas testing is done on the test data. Additional default hyperparameters are displayed in table II. The results of experiment 5 are displayed in table VII. Examples of spike patterns and learned weights are shown in figure 14.

2) Interpretation: Experiment 5.1a, which is included as a sanity check, shows how without training the network achieves an accuracy of 9.79%, which (as it should) practically matches chance performance (9.09%). Experiments 5.1b-c show how increasing the number of train cycles beyond 3 does not greatly improve network performance. Experiment 5.2 shows how having too few circuits in layer $\mathcal{L}_1$ decreases performance significantly. And experiment 5.3 shows that having too few neurons in each circuit hurts performance. Finally, experiment 5.7c shows that the combination of the optimal parameters of experiments 5.1-5.3 yields a — within this work — optimal performance of 72.69%.

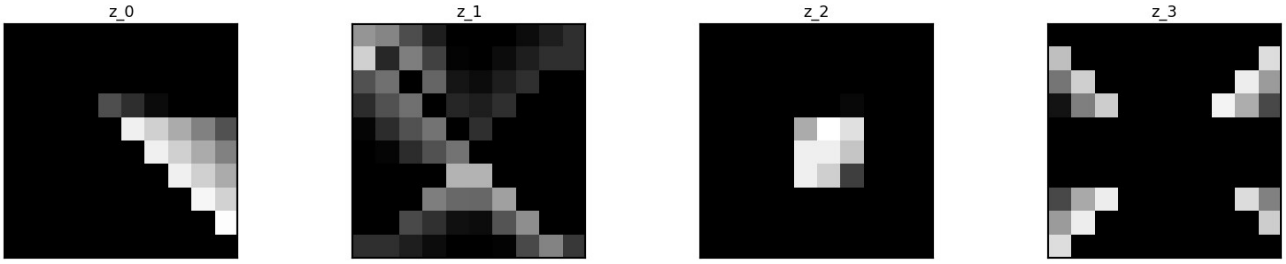
It makes sense that increasing the the complexity of the network, with respect to the above hyperparameters, improves performance. Experiments 5.4-5.7, however, show that network performance is impacted in less predictable ways by a different selection of hyperparameters. For example, it is less easy to predict that setting learning rate decay $\hat{\eta}$ to 1.00 decreases performance by over 15% relative to when it is set to the default of 0.60. Furthermore, it is not straightforward to decide why omitting the MFCCs hurts performance by roughly 10%, while omitting their deltas hurts it by nearly 30%. All that can be said at this point, is that these hyperparameters impact performance significantly, though the exact manner in



(a) Spikes produced by the first (and only) circuit of layer $\mathcal{L}_1$.



(b) Output spike produced by the single circuit of layer $\mathcal{L}_2$.



(c) Visualization of the **gate** connection weights of the first (and only) circuit of layer $\mathcal{L}_1$ following the train phase of experiment 2.1c. It shows how stochastic neurons might not produce a spike at the exact time that the sub-patterns end, and thus how they may learn to recognize sub-optimal patterns.

Fig. 12. Visualizations corresponding to experiment 2.3a. Spikes of the sensory layer (for stimulus 40 == stimulus 4) are displayed in figure 10a.

which they do so is not entirely clear.

In experiment 5.7b the neuron connection weights $\boldsymbol{w}$ are again allowed to evolve according to STDP dynamics. As the results show, this hurts performance by roughly 6%. It should be noted that, with better fine-tuning, learning in the neuron connection weights might prove beneficial. However, gate connection weights $\boldsymbol{\omega}$, rather than neuron connection weights $\boldsymbol{w}$, are the focus of this work. Thus, in order to avoid the extra hyperparameters and fine-tuning, the neuron connection weights were fixed to 1.0 throughout most of this work. More attention can be paid to this subject in future work.

It should be noted that the above observations are limited by the fact that the impact of the variation of one hyperparameter may depend on the settings of other hyperparameters. For example, it makes sense that increasing the number of circuits and neurons will require more train cycles to reach optimal performance. Likewise, learning rate decay $\hat{\eta}$ may influence the optimal number of train cycles, or vice versa. These kinds of interactions make it difficult to ascertain the optimal network settings. Deeper understanding of the algorithm, its dynamics, and its behaviour inform such choices, and furthering such understanding should be a focus of future work.

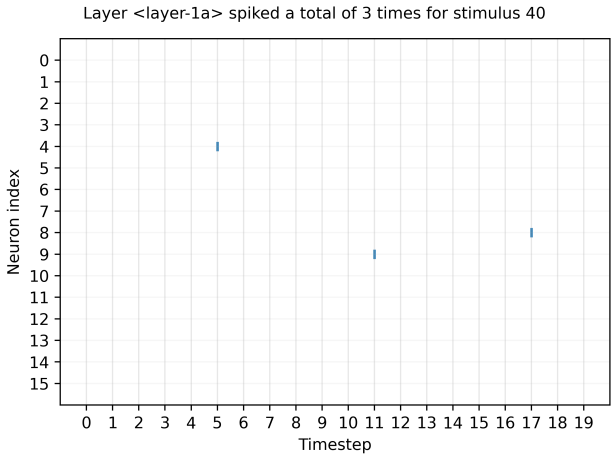
V. Discussion

The experiments have provided a number of insights into the capabilities of WTA circuits. Principal among these insights are as follows. First of all, experiment 1 shows that static WTA circuits are not capable of distinguishing between time-varying spiking patterns, while temporal WTA circuits are. Secondly, experiment 2 shows that temporal WTA circuits struggle when patterns start to repeat within the timescale at which they operate. It further shows that this problem can be solved by forming multiple circuits into a network, where different layers operate at different timescales. Thirdly, experiment 3 shows how a two-layer temporal WTA network can learn to perfectly remember utterances of a single speaker. Finally, experiments 4 and 5 show that the two-layer temporal WTA network is capable of learning to distinguish between spoken digits.

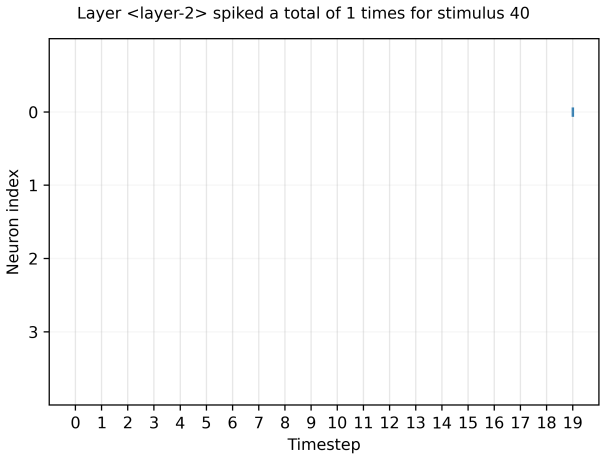
A. Sparsity

While the temporal WTA circuit is more versatile than the static WTA circuit, it comes at the cost of a large increase in number of learnable parameters. Specifically, where the static WTA circuit has $K \cdot N$ learnable parameters $\boldsymbol{w}$ (which, as shown in the experiments, can be omitted in the temporal WTA circuit), the temporal WTA circuit has $K \cdot N^2$ learnable parameters $\boldsymbol{\omega}$. The experiments show however, that the large majority of weights $\boldsymbol{\omega}$ equal zero after learning, at which point they can be omitted entirely.

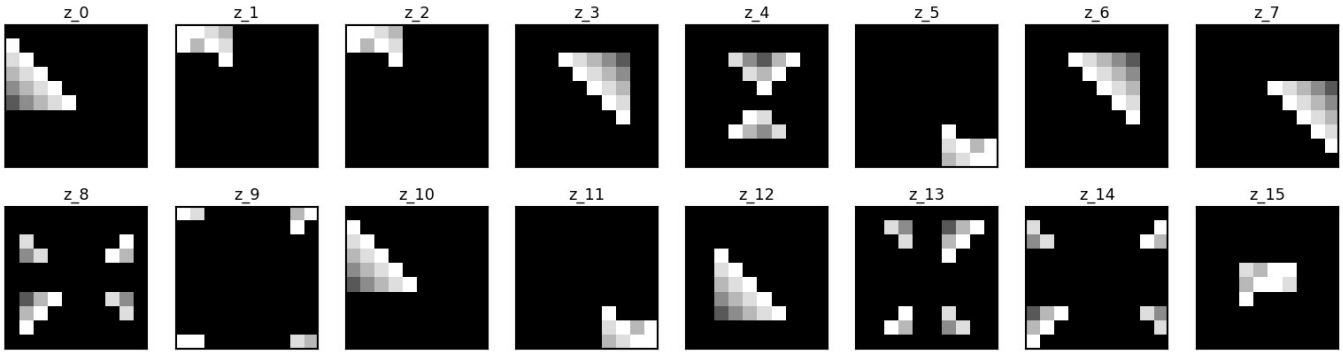
Several observations can be made about the sparsity of temporal WTA circuits following training. First of all, the contrast between experiment 1.2a and 1.2b shows that an increased number of neurons N does not necessarily increase the amount of non-zero weights by a large amount. Experiment



(a) Spikes produced by the first circuit of layer $\mathcal{L}_1$, in experiment 2.1d there are 4 more circuits in $\mathcal{L}_1$ that generate similar spike patterns. Together, this averages away the randomness inherent to stochastic neurons.



(b) Output spike produced by the single circuit of layer $\mathcal{L}_2$.



(c) Visualization of the **gate** connection weights of the first (and only) circuit of layer $\mathcal{L}_1$ following the train phase of experiment 2.1d. It shows how by increasing the number of stochastic neurons, the desired pattern can be pieced together.

Fig. 13. Visualizations corresponding to experiment 2.3b. Spikes of the sensory layer (for stimulus 40 == stimulus 4) are displayed in figure 10a.

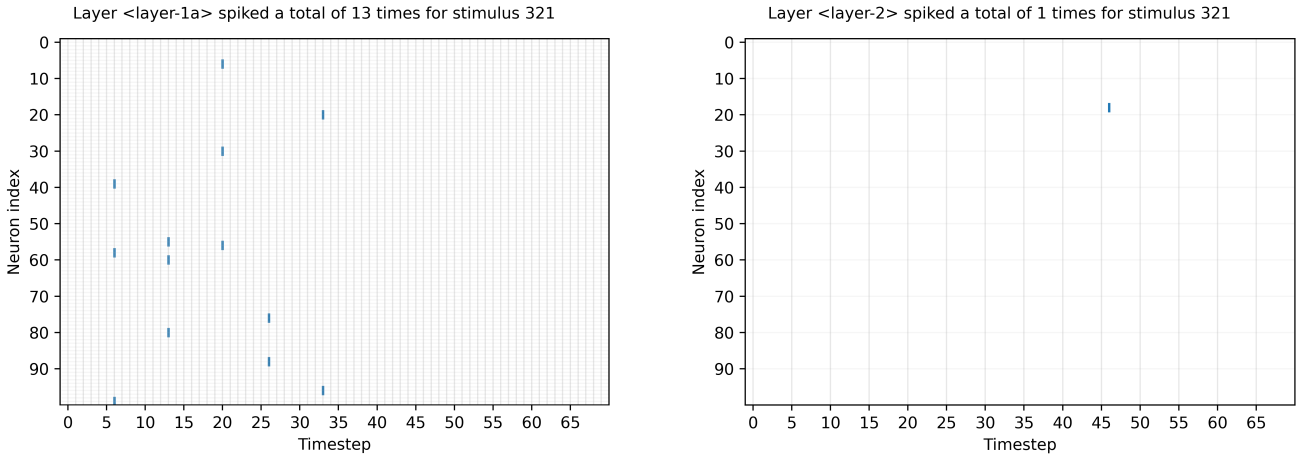
1.2a has $N = 100$ input neurons while experiment 1.2b has $N = 10$. In both experiments however, only 10 neurons spike for a given stimulus. Thus, while experiment 1.2a has 100 times as many learnable parameters as experiment 1.2b, following learning it has less than thrice as many non-zero weights. Contrary to this, experiment 5.1c has roughly four times as many learnable parameters as experiment 5.6a, as well as roughly four times as many non-zero weights after learning. In this case, the sparsity of the input is the same, and thus the sparsity of the weights is roughly the same.

Thus, while the number of non-zero weights grows as the number of input neurons increases, this growth is for a large part mitigated if the input neurons fire sparsely. Given that WTA circuits produce sparse spiking patterns by design, this is all but guaranteed in subsequent layers of a WTA network. This is shown in experiment 5, where in most experiments less than 1% of the weights of layer $\mathcal{L}_2$ are non-zero after training. On the appropriate neuromorphic device, the sparsity of temporal WTA circuits will eliminate much of the complexity of the algorithm. Future work can delve more into this topic, exploring for example how sparsity might be increased further, or exploring whether sparsity can be introduced even before training.

B. Network development

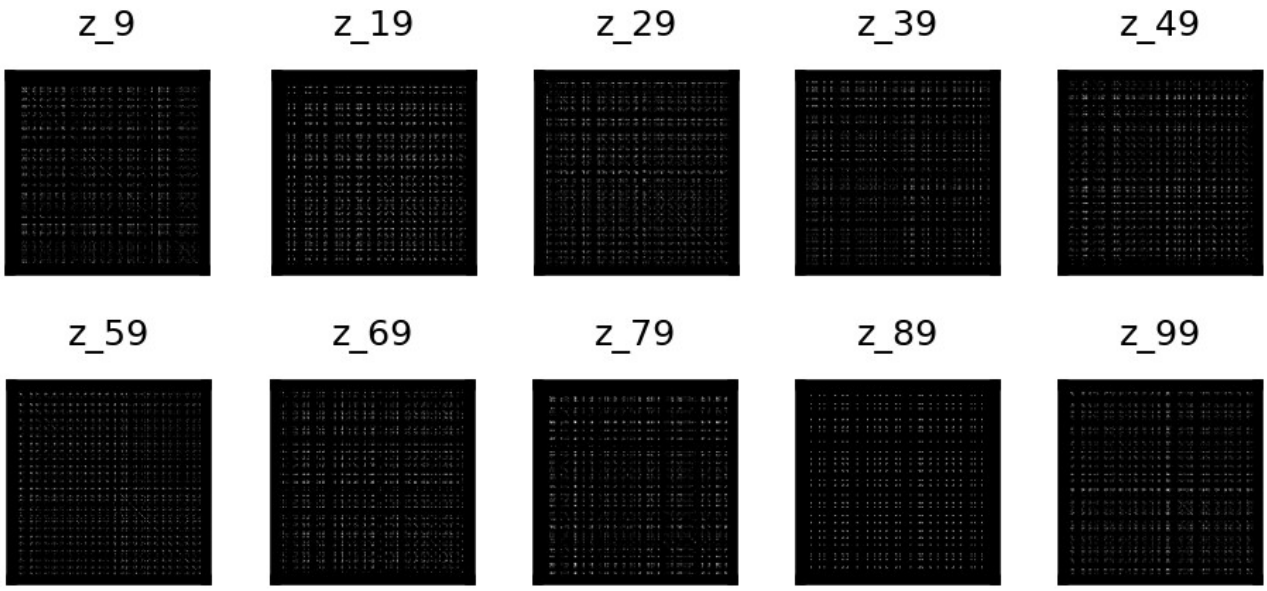
On the one hand, the temporal WTA network can be further developed by increasing its complexity. It can be developed by increasing the complexity of components, such as by adopting more complex neuron models. It can be developed by increasing the complexity of network structure, such as by adopting hierarchical structures as are employed by [42]. Furthermore, it is possible to add altogether new types of components to the network, such as explicitly modelled axons, synapses, and dendrites. This offers a broad range of possibilities for future work to improve the algorithm.

On the other hand, the temporal WTA network can be further developed by decreasing its complexity. The two-layer architecture already has many hyperparameters, and as touched on in experiment 5, it is not straightforward to find an optimal setting for these parameters. Introducing more hyperparameters will make this task more difficult. Instead, future work can focus on refining existing dynamics. The refining of existing dynamics may ensure that the impact of each hyperparameter is better understood, or may cause certain hyperparameters to be eliminated entirely. Where this fails, methods such as evolutionary algorithms can be employed to optimize hyperparameters.



(a) Spikes produced by the first circuit of layer $\mathcal{L}_1$ of a network trained on the entire TIDIGITS dataset.

(b) Output spike produced by the single circuit of layer $\mathcal{L}_2$.



(c) A selection of 10 visualizations of **gate** connection weights of the first circuit of layer $\mathcal{L}_1$ following the training on the TIDIGITS dataset.

Fig. 14. Visualization of spikes and weights of a default two-layer network with $K_2 = 100$ softmax neurons in layer $\mathcal{L}_2$, trained on the TIDIGITS data. Spikes of the sensory layer are displayed in figure 7b.

A different way of developing the algorithm, that does not necessarily increase or decrease its complexity, is implementing it in a more neuromorphic fashion. By implementing the algorithm via simulation on a von Neumann machine, it is limited in several ways. First of all, it is limited to discrete time, thereby sacrificing temporal resolution. Secondly, it is not event based. Instead, each timestep processes each operation in a pre-determined sequential order. The expectation is that, given the appropriate neuromorphic device, the network dynamics can be adapted to unfold in continuous time[6], fully event-based, and fully parallel. This is another avenue for future work to explore.

[6]Note that in [40], [41] the WTA circuit dynamics were conceived as part of a continuous time model.

C. Input encoding

The encoding of the TIDIGITS utterances used in this work is heavily based on techniques that have been shown to thrive in the field of deep learning, specifically, the use of MFCCs. Encoding MFCCs by associating neurons with binned MFCC values is not a brain-inspired approach, but it has several properties that are desirable for this work. First of all, it is a very straightforward translation of MFCCs to spikes. Some information is lost by binning the MFCCs, but overall the spike patterns very closely resemble the MFCCs. This ensures that the spiking patterns contain the information necessary to distinguish between digits. Secondly, while a stimulus lasts, each timestep yields an identical amount of input spikes (one for each coefficient).

The stability of this encoding makes balancing network excitation easier. For example, this avoids the problem of ex-

ploding membrane potentials, which might occur if stochastic neurons receive excitation in large bursts. As experiment 5.6 shows, variations in the input can have a large impact on network performance. In future work it will be interesting to explore the impact of different encodings.

D. Softmax and stochastic neurons

Throughout this work both stochastic and softmax neurons are used. As touched on before, softmax neurons violate various neuromorphic principles. The steady pre-determined interval at which a layer of softmax neurons produces spikes requires global moderation. Furthermore, the computation of the softmax function over membrane potentials requires that each softmax neuron has knowledge of the membrane potentials of all others in the same circuit.

Stochastic neurons provide a neuromorphic alternative to softmax neurons. As shown in [41], when combined with the competition mechanisms of a WTA circuit, it holds that, at each moment in time that a stochastic neuron spikes, the combined spiking probabilities of the neurons follow a softmax distribution over their membrane potentials. This holds only in continuous time, where the probability of exactly simultaneous spikes is zero[7].

In a discrete time implementation it is possible for multiple stochastic neurons to spike at the exact same time. Particularly during the initial stages of learning this can cause problems. At this stage, neurons have not yet grown distinctly sensitive to specific stimuli, risking the simultaneous spiking of many neurons, which consequently grow sensitive to the same patterns. A more elegant solution might be to not initialize all neurons and connections weights at the same time with high weights. Instead, the initial phases of learning might involve the gradual genesis of neurons and connections. Such a solution might also ensure sparsity even from the very start. This, and other solutions, are left for future work to explore.

E. Separation of stimuli

In this work, the start and end of each stimulus is accompanied by non-neuromorphic operations. At the onset of each stimulus, variables such as membrane potentials are reset to their resting state. At the end of each stimulus, a single spike is produced an output layer of softmax neurons. In a realistic setting stimuli are ever-present and overlapping. In such settings, information about the start and end of a stimulus is not straightforwardly available, nor is it directly accessible for each individual component. Instead, a more realistic approach would be to have separate SNNs dedicated for processes such as attention and segmentation, and to have these communicate this information via neuron connections.

While parts of our method are un-neuromorphic in this area, others are not. Specifically, it should be noted that layer $\mathcal{L}_1$ of stochastic neurons has no knowledge of what it is trying to distinguish between, and has no outside help in determining the onset or end of specific patterns (save at the onset of the larger stimulus). Experiment 2 shows how this can cause stochastic neurons to learn sub-optimal patterns, but that by increasing the amount of neurons and circuits this noise can be averaged out. In this fashion, stochastic neurons are able to distinguish between patterns well enough to provide useful information for softmax layer $\mathcal{L}_2$. In a more realistic setting, there will be no layers of softmax neurons, but only layers of stochastic neurons, which interact with other networks responsible for things such as action and attention. This leaves a wide range of possibilities for future work to explore.

[7]Note that, while the probability of exactly simultaneous spikes is zero in continuous time, multiple spikes may still follow on another in very quick succession (at speeds too quick for lateral inhibition signals to interfere).

F. Relation to other works

In line with previous research on WTA circuits [41]–[43], this work employs accuracy as its performance measure. In previous work, the accuracy was measured with respect to the statically encoded MNIST dataset, which concerns 10 handwritten digit classes. In this research, it was measured with respect to the temporally encoded TIDIGITS dataset, which concerns 11 spoken digit classes. Given that WTA circuits learn entirely via unsupervised processes, a mapping is decided following learning in order to be able to compute the accuracy. In future work it would instead be interesting to include a supervised component. For example, layer $\mathcal{L}_2$ of the two-layer network could be replaced with a tempotron, which can use supervised learning to interpret the spikes of layer $\mathcal{L}_1$. It would also be interesting to explore the inclusion of reinforcement or supervised learning signals, and the direct integration of these with the existing weight updates rules.

This work has not made any direct comparisons to other works with respect to performance. The primary reason for this is that the goal of this work is not to introduce a new state-of-the-art speech recognition algorithm, but to provide the foundation for a truly neuromorphic algorithm. In this light, the temporal WTA circuit distinguishes itself from related work in various ways. The algorithm distinguishes itself from the tempotron by being a fully unsupervised algorithm that can process at various timescales by combining multiple circuits into a network. It distinguishes itself from the LSM by distinguishing between spike patterns, rather than merely echoing them. And it distinguishes itself from the SOM by processing input locally and in real time.

In essence, we present the temporal WTA circuit as having the potential to be a truly neuromorphic approach, capable of satisfying all of the neuromorphic principles discussed in section I. As discussed in detail in earlier sections, this claim does not hold for the current implementation of the temporal WTA circuit. Even so, as has been addressed in each respective section, we argue that the limitations of the current implementation are not fundamental limitations of the method itself.

VI. Conclusion

This work introduces a novel neuromorphic approach for processing time-varying stimuli: the temporal WTA circuit. The temporal WTA circuit differentiates itself from the traditional (static) WTA circuit by being able to distinguish between time-varying (rather than just static) spike patterns.

The temporal WTA circuit is able to learn such distinctions between patterns in an online, real-time, and unsupervised manner. This work has shown that temporal WTA circuits can be combined into networks to process more complex stimuli, using multiple layers that process input at different timescales. Furthermore, this work has shown that this allows the network to learn to distinguish between the spoken digits of the TIDIGITS dataset. While several details of the network implementation in this work violate neuromorphic principles, these are addressed throughout this work, and it is argued that these do not extend to the method in general.

References

[1] G. E. Moore, "Cramming more components onto integrated circuits, reprinted from electronics, volume 38, number 8, april 19, 1965, pp.114 ff," *IEEE Solid-State Circuits Society Newsletter*, vol. 11, no. 3, pp. 33–35, April 2006. [Online]. Available: 10.1109/N-SSC.2006.4785860

[2] A. Sanchez, S. Bertolazzi, and P. Cambou, "Neuromorphic computing and sensing 2021," *Yole Développement*, May 2021.

[3] C. Shipley and S. M. Jodis, "Programming languages classification," *Encyclopedia of Information Systems*, pp. 545–552, 2002.

[4] J. Backus, "Can programming be liberated from the von neumann style? a functional style and its algebra of programs," *Communications of the ACM*, vol. 21, no. 8, pp. 613–641, August 1978. [Online]. Available: 10.1145/359576.359579

[5] S. Herculano-Houzel, "The remarkable, yet not extraordinary, human brain as a scaled-up primate brain and its associated cost," *Proceedings of the National Academy of Sciences of the United States of America*, June 2012. [Online]. Available: https://doi.org/10.1073/pnas.1201895109

[6] G. Buzsáki, N. Logothetis, and W. Singer, "Scaling brain size, keeping timing: Evolutionary preservation of brain rhythms," *Neuron*, vol. 80, no. 3, pp. 751–764, October 2013. [Online]. Available: 10.1016/j.neuron.2013.10.002

[7] T. Branco and K. Staras, "The probability of neurotransmitter release: variability and feedback control at single synapses," *National Library of Medicine*, vol. 10, no. 5, pp. 373–383, May 2009. [Online]. Available: 10.1038/nrn2634

[8] S. Furber, "Large-scale neuromorphic computing systems," *Journal of Neural Engineering*, vol. 13, no. 5, p. 051001, August 2016. [Online]. Available: 10.1088/1741-2560/13/5/051001

[9] C. Schuman, S. Kulkarni, M. Parse, J. Mitchell, P. Date, and B. Kay, "Opportunities for neuromorphic computing algorithms and applications," *Nature Computational Science*, vol. 2, no. 1, pp. 10–19, January 2022. [Online]. Available: 10.1088/2634-4386/ac889c

[10] D. Christensen, R. Dittman, B. Linares-Barranco, A. Sebastian, M. Le Gallo, A. Redaelli, S. Slesazeck, T. Mikolajick, S. Spiga, S. Menzel, I. Valov, G. Milano, C. Ricciardi, S.-J. Liang, F. Miao, M. Lanza, T. Quill, S. Keene, A. Salleo, J. Grollier, D. Markovi, A. Mizrahi, P. Yao, J. Yang, G. Indiveri, J. Strachan, S. Datta, E. Vianello, A. Valentian, J. Feldmann, X. Li, W. Pernice, H. Bhaskaran, S. Furber, E. Neftci, F. Scherr, W. Maass, S. Ramaswamy, J. Tapson, P. Priyadarshini, Y. Kim, G. Tanaka, S. Thorpe, C. Bartolozzi, T. Cleland, C. Posch, S. Liu, G. Panuccio, M. Mahmud, A. Mazumder, M. Hosseini, T. Mohsenin, E. Donati, S. Tolu, R. Galeazzi, M. Christensen, S. Holm, D. Ielmini, and N. Pryds, "2022 roadmap on neuromorphic computing and engineering," *Neuromorphic Computing and Engineering*, vol. 2, no. 2, p. 022501, May 2022. [Online]. Available: 10.1088/2634-4386/ac4a83

[11] S.-C. Liu and T. Delbruck, "Neuromorphic sensory systems," *Current Opinion in Neurobiology*, vol. 20, no. 3, pp. 288–295, June 2010. [Online]. Available: 10.1016/j.conb.2010.03.007

[12] M. Alawad and M. Lin, "Survey of stochastic-based computation paradigms," vol. 7, no. 1, pp. 98–114, 2019. [Online]. Available: 10.1109/TETC.2016.2598726

[13] J. C. Gallagher, "The once and future analog alternative: evolvable hardware and analog computation," *NASA/DoD Conference on Evolvable Hardware, 2003. Proceedings*, pp. 43–49, 2003. [Online]. Available: 10.1109/EH.2003.1217641

[14] F. Ogban, I. Arikpo, and I. Eteng, "Von neumann architecture and modern computers," *Global Journal of Mathematical Sciences*, vol. 6, no. 2, pp. 97–104, September 2007. [Online]. Available: 10.4314/gjmas.v6i2.21415

[15] J. Aimone, P. Date, G. Fonseca-Guerra, K. Hamilton, K. Henke, B. Kay, G. Kenyon, S. Kulkarni, S. Mniszewski, M. Parsa, S. Risbud, C. Schuman, W. Severa, and J. D. Smith, "A review of non-cognitive applications for neuromorphic computing," *Neuromorphic Computing and Engineering*, vol. 2, no. 3, p. 032003, 09 2022. [Online]. Available: 10.1088/2634-4386/ac889c

[16] L. Deng, "Artificial intelligence in the rising wave of deep learning: The historical path and future outlook [perspectives]," *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 180–177, January 2018. [Online]. Available: 10.1109/MSP.2017.2762725

[17] Z. Shao, R. Zhao, S. Yuan, M. Ding, and Y. Wang, "Tracing the evolution of ai in the past decade and forecasting the emerging trends," *Expert Systems with Applications*, vol. 209, no. 15, p. 118221, December 2022. [Online]. Available: 10.1016/j.eswa.2022.118221

[18] R. Wason, "Deep learning: Evolution and expansion," *Cognitive Systems Research*, vol. 52, pp. 701–708, December 2018. [Online]. Available: 10.1016/j.cogsys.2018.08.023

[19] P. U. Diehl, G. Zarella, A. Cassidy, B. U. Pedroni, and E. Neftci, "Conversion of artificial recurrent neural networks to spiking neural networks for low-power neuromorphic hardware," *2016 IEEE International Conference on Rebooting Computing (ICRC)*, pp. 1–8, October 2016. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/ICRC.2016.7738691

[20] E. Hunsberger and C. Eliasmith, "Training spiking deep networks for neuromorphic hardware," November 2016. [Online]. Available: 10.13140/RG.2.2.10967.06566

[21] W. Severa, C. M. Vineyard, R. Dellana, S. J. Verzi, and J. B. Aimone, "Training deep neural networks for binary communication with the whetstone method," *Nature Machine Intelligence*, vol. 1, no. 2, pp. 86–94, February 2019. [Online]. Available: 10.1038/s42256-018-0015-y

[22] J. H. Lee, T. Delbruck, and M. Pfeiffer, "Training deep spiking neural networks using backpropagation," *Frontiers in Neuroscience*, vol. 10, November 2016. [Online]. Available: 10.3389/fnins.2016.00508

[23] A. Shrestha, H. Fang, Q. Wu, and Q. Qiu, "Approximating back-propagation for a biologically plausible local learning rule in spiking neural networks," *Proceedings of the International Conference on Neuromorphic Systems*, vol. 10, pp. 1–8, July 2019. [Online]. Available: 10.1145/3354265.3354275

[24] D. Kwon, S. Lim, J.-H. Bae, S.-T. Lee, H. Kim, Y.-T. Seo, S. Oh, J. Kim, K. Yeom, B.-G. Park, and J.-H. Lee, "On-chip training spiking neural networks using approximated backpropagation with analog synaptic devices," *Frontiers in Neuroscience*, vol. 14, July 2020. [Online]. Available: 10.3389/fnins.2020.00423

[25] G. Bellec, F. Scherr, A. Subramoney, E. Hajek, D. Salaj, R. Legenstein, and W. Maass, "A solution to the learning dilemma for recurrent networks of spiking neurons," *Nature Communications*, vol. 11, no. 1, p. 3625, July 2020. [Online]. Available: 10.1038/s41467-020-17236-y

[26] R. Gütig and H. Sompolinsky, "The tempotron: a neuron that learns spike timing–based decisions," *Nature Neuroscience*, vol. 9, no. 3, pp. 420–428, March 2006. [Online]. Available: 10.1038/nn1643

[27] W. Maass, T. Natschläger, and H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," *Neural Computation*, vol. 14, no. 11, pp. 2531–2560, November 2002. [Online]. Available: 10.1162/089976602760407955

[28] H. Jaeger, "The "echo state" approach to analysing and training recurrent neural networks – with an erratum note," January 2001. [Online]. Available: https://www.ai.rug.nl/minds/uploads/EchoStatesTechRep.pdf

[29] M. Cucchi, S. Abreu, G. Ciccone, D. Brunner, and H. Kleemann, "Hands-on reservoir computing: a tutorial for practical implementation," *Neuromorphic Computing and Engineering*, vol. 2, no. 3, August 2022. [Online]. Available: 10.1088/2634-4386/ac7db7

[30] D. Neil and S.-C. Liu, "Effective sensor fusion with event-based sensors and deep network architectures," *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 2282–2285, 2016. [Online]. Available: 10.1109/ISCAS.2016.7539039

[31] M.-H. Tayarani-Najaran and M. Schmuker, "Event-based sensing and signal processing in the visual, auditory, and olfactory domain: A review," *Frontiers in Neural Circuits*, vol. 15, May 2021. [Online]. Available: 10.3389/fncir.2021.610446

[32] G. Gallego, T. Delbrück, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. J. Davison, J. Conradt, K. Daniilidis, and D. Scaramuzza, "Event-based vision: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 1, pp. 154–180, July 2022. [Online]. Available: 10.1109/TPAMI.2020.3008413

[33] L. Deckers, I. J. Tsang, W. Van Leekwijck, and S. Latré, "Extended liquid state machines for speech recognition," *Frontiers in Neuroscience*, vol. 16, October 2022. [Online]. Available: 10.3389/fnins.2022.1023470

[34] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990. [Online]. Available: 10.1109/5.58325

[35] S. Zeki, "The representation of colours in the cerebral cortex," *Nature*, vol. 284, pp. 412–418, 1980. [Online]. Available: 10.1038/284412a0

[36] M. Saenz and D. R. M. Langers, "Tonotopic mapping of human auditory cortex," *Hearing Research*, vol. 307, pp. 42–52, January 2014. [Online]. Available: 10.1016/j.heares.2013.07.016

[37] H. P. Killackey, R. W. Rhoades, and C. A. Bennett-Clarke, "The formation of a cortical somatotopic map," *Trends in Neuroscience*, vol. 18, no. 9, pp. 402–407, September 1995. [Online]. Available: 10.1016/0166-2236(95)93937-S

[38] J. Wu, Y. Chua, and H. Li, "A biologically plausible speech recognition framework based on spiking neural networks," *2018 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2018. [Online]. Available: 10.1109/IJCNN.2018.8489535

[39] D. Yu, M. L. Seltzer, J. Li, J. T. Huang, and F. Seide, "Feature learning in deep neural networks - studies on speech recognition tasks," 2013. [Online]. Available: 10.48550/arXiv.1301.3605

[40] B. Nessler, M. Pfeiffer, and W. Maass, "Stdp enables spiking neurons to detect hidden causes of their inputs," *Advances in Neural Information Processing Systems (NIPS)*, vol. 22, pp. 1357–1365, January 2009.

[41] B. Nessler, M. Pfeiffer, L. Buesing, and W. Maass, "Bayesian computation emerges in generic cortical microcircuits through spike-timing-dependent plasticity," *PLOS Computational Biology*, vol. 9, no. 4, pp. 1–30, May 2013. [Online]. Available: 10.1371/journal.pcbi.1003037

[42] S. Guo, Z. Yu, F. Deng, X. Hu, and F. Chen, "Hierarchical bayesian inference and learning in spiking neural networks," *IEEE Transactions on Cybernetics*, vol. 49, no. 1, pp. 133–145, January 2019. [Online]. Available: 10.1109/TCYB.2017.2768554

[43] O. van der Himst, L. Bagheriye, and J. Kwisthout, "Bayesian integration of information using top-down modulated winner-take-all networks," August 2023. [Online]. Available: 10.48550/arXiv.2308.15390

[44] R. G. Leonard and G. Doddington, "Tidigits speech corpus," *Linguistic Data Consortium, Philadelphia*, 1993.

VII. Appendix A – Notation

TABLE I. Notation

Miscellaneous:	
α	Locally defined constant
β	Locally defined constant
$f(...)$	Locally defined function
B	number of bins used to encode each MFCC
Static WTA circuit:	
$\boldsymbol{z}$	WTA circuit neurons $\{z_1, ..., z_K\}$
$\boldsymbol{y}$	input neurons $\{y_1, ..., y_N\}$ that encode input as a spatiotemporal spike pattern
$\boldsymbol{c}$	neuron connections $\{c_{kn} \vert k \in \{1, ..., K\}, n \in \{1, ..., N\}\}$ that allow spikes to travel from neurons $\boldsymbol{y}$ to neurons $\boldsymbol{z}$
t	discrete variable indicating a specific point in time
T	duration of each stimulus in terms of timesteps (assumes duration is the same for all stimuli)
T_x	duration of stimulus x in terms of timesteps
$I(t)$	lateral inhibition signal that inhibits membrane potentials $\boldsymbol{\mu}$ of neurons $\boldsymbol{z}$ after a spike from a neuron z_k
$\boldsymbol{w}(t)$	neuron connection weights $\{w_{kn}(t) \vert k \in \{1, ..., K\}, n \in \{1, ..., N\}\}$ that weigh spikes travelling through connections $\boldsymbol{c}$, initialized at random and restricted to the range $[0,\ 1]$
$\boldsymbol{\mu}(t)$	membrane potentials $\{\mu_1(t), ..., \mu_k(t)\}$ of neurons $\boldsymbol{z}$, initialized at μ^{min} and restricted to the range $[0,\ \mu^{max}]$
$\phi_n(t)$	equals 1 if neuron y_n spiked at time t, and 0 otherwise
$\bar{\phi}_n(t)$	equals 0 if neuron y_n spiked at time t, and 1 otherwise
$\zeta_k(t)$	equals 1 if neuron z_k spiked at time t, and 0 otherwise
$\bar{\zeta}_k(t)$	equals 0 if neuron z_k spiked at time t, and 1 otherwise
$\zeta(t)$	equals 1 if any neuron in $\boldsymbol{z}$ spiked at time t, and 0 otherwise
$\bar{\zeta}(t)$	equals 0 if any neuron in $\boldsymbol{z}$ spiked at time t, and 1 otherwise
t^{Δ}_{kn}	scalar variable indicating the temporal distance between the most recent spikes of neurons z_k and y_n at time t
$\eta_k(t)$	adaptive learning rate that diminishes each time neuron z_k spikes according to equation 10
$\hat{\eta}$	Influences the speed at which each learning rate $\eta_k(t)$ decays according to 10.
Temporal WTA circuit:	
$\boldsymbol{\gamma}$	neuron connection gates $\{\gamma_{kn} \vert k \in \{1, ..., K\}, n \in \{1, ..., N\}\}$ attached to each neuron connection
$\boldsymbol{\theta}(t)$	gate conductances $\{\theta_{kn}(t) \vert k \in \{1, ..., K\}, n \in \{1, ..., N\}\}$ that, in addition to weights $\boldsymbol{w}$, weigh spikes travelling through connections $\boldsymbol{c}$, initialized at 0 and restricted to the range $[0,\ \infty]$
τ	discrete constant that indicates the timespan at which gates $\boldsymbol{\gamma}(t)$ operate
$\boldsymbol{\kappa}$	gate connections $\{\kappa_{knn'} \vert k \in \{1, ..., K\}, n \in \{1, ..., N\}, n' \in \{1, ..., N\}\}$ that allow gates to influence the conductance of other gates
$\boldsymbol{\omega}(t)$	gate connection weights $\{\omega_{knn'}(t) \vert k \in \{1, ..., K\}, n \in \{1, ..., N\}, n' \in \{1, ..., N\}\}$ that weigh the degree to which each gate perturbs the conductance of other gates, initialized at random and restricted to the range $[0,\ 1]$
$\boldsymbol{\rho}(t)$	recency traces $\{\rho_{knn'}(t) \vert k \in \{1, ..., K\}, n \in \{1, ..., N\}, n' \in \{1, ..., N\}\}$ that represent the temporal distance between the most recent spikes of neurons y_n and y'_n, initialized at 0 and restricted to the range $[0,\ \tau]$
$\boldsymbol{\pi}(t)$	prime traces $\{\pi_{knn'}(t) \vert k \in \{1, ..., K\}, n \in \{1, ..., N\}, n' \in \{1, ..., N\}\}$ that are a measure of how closely and how consistently a spike from neuron $y_{n'}$ was preceded by a spike from neuron y_n, initialized at 0 and restricted to the range $[0,\ \infty]$
Two-layer temporal WTA network:	
$\mathcal{L}_1$	WTA layer one, consists of $\mathcal{N}_1$ temporal WTA circuits that each receive input from all sensory neurons $\boldsymbol{y}$; by default $\mathcal{L}_1$ consists of stochastic neurons
$\mathcal{L}_2$	WTA layer two, consists of a single temporal WTA circuit that receives input from all neurons in layer $\mathcal{L}_1$, the single output spike of $\mathcal{L}_2$ at the end of a stimulus is used to determine the network's classification

VIII. Appendix B – Hyperparameters

TABLE II. Default hyperparameters of experiments 3, 4, and 5

Variable	Code name	Default	Description
-	`data_form`	MFCCs & deltas	As described in section IV-A3, when encoding the TIDIGITS data one can use the MFCCs, their deltas, or both. By default both are used.
B	`n_bins`	8	As described in section IV-A3, when encoding the TIDIGITS data the values of each coefficient are divided into a number of bins, each bin being assigned a sensory neuron.
$\mathcal{N}_1$	`l1_n_circuits`	5	As described in section IV-B, WTA layer $\mathcal{L}_1$ consists of $\mathcal{N}_1$ temporal WTA circuits. The impact of this parameter is assessed in experiment 5.2.
K_1	`l1_K`	100	The number of stochastic neurons in each circuit of WTA layer $\mathcal{L}_1$. The impact of this parameter is assessed in experiments 3.3, 4.3, and 5.3.
-	`l1_hz`	150	Determines the timescale at which WTA layer $\mathcal{L}_1$ operates: if equal to 150 then for layer $\mathcal{L}_1$ it holds that $\tau = \frac{1000}{150}$.
-	`l1_n_max_simultaneous_spikes`	3	The maximum allowed number of simultaneous spikes produced within circuits of WTA layer $\mathcal{L}_1$. The impact of this parameter is assessed in experiment 5.5 and discussed in section V.
α (eq. 4)	`l1_cst_p_spike`	30	Influences how quickly the spiking probability of stochastic neurons rises as a function of their membrane potential. A higher value means that the spike probability of a neuron starts to rise rapidly more nearer to its maximum membrane potential μ^{max}. A very low value introduces too much randomness, while a high value will eliminate it. See figure 3a.
μ^{max}	`l1_mps_max`	1500	The maximum membrane potential of stochastic neurons. When a stochastic neuron's membrane potential equals μ^{max} its spiking probability equals 1. The default value of 1500 is chosen to fit roughly with `l1_hz`, which requires that a stochastic WTA circuit produces a spike roughly every $\frac{1000}{150}$ timesteps.
K_2	`l2_K`	Varies	The number of neurons in the temporal WTA circuit that is layer $\mathcal{L}_2$. After training each neuron is associated with a class, thus there should be at least as many neurons as there are classes. Adding more neurons than classes is necessary to account for within-class variation. The impact of this parameter is assessed in experiments 3.3 and 5.3.
-	`l2_idle_until`	Varies	The number of stimuli for which WTA layer $\mathcal{L}_2$ is idle, allowing WTA layer $\mathcal{L}_1$ to first learn and stabilize somewhat. By default $\mathcal{L}_2$ is idle for the first 0.60 train cycles, rounded down.
-	`l2_hz`	20	Determines the timescale at which WTA layer $\mathcal{L}_1$ operates: if equal to 20 then for layer $\mathcal{L}_2$ it holds that $\tau = \frac{1000}{20}$. The default setting thus assumes to operate roughly a time windows of 50 timesteps, which reflects (very roughly) the duration of the single-digit utterances (which vary from 24 to 70 timesteps).
-	[`s_w_init_min`, `s_w_init_max`]	$[0.60, 0.80]$	The initial range of values at which gate connection weights are initialized. Chosen to be relatively high such that each neuron is likely to respond and grow sensitive to a spiking pattern at least once.
$\hat{\eta}$	`eta_decay`	0.60	As described in section III-B, $\hat{\eta}$ influences the speed at which learning rate $\eta_k(t)$ decays. The impact of this parameter is assessed in experiment 5.4.
-	`eta_star`	25	For the temporal WTA circuits, when learning is triggered, the weight update is repeated `eta_star` times (each weight update being recomputed according to the new weights, and η decaying each time). This is a bit of a hack to increase the rate of learning without violating certain dynamics (for example, initializing η at 10 instead of 1 causes weight updates to act strangely). A more elegant solution is likely possible.
-	`axon_en_learn`	False	If False, then all neuron connection weights are fixed at 1.0, and they do not change at all during training.

IX. Appendix C – Experiment Results

TABLE III. Results experiment 1

ID	Custom parameters	Avg. acc	Stdv. acc	Sparsity $\mathcal{L}_1$	Sparsity $\mathcal{L}_2$
1.1a	– Static WTA circuit – Static encoding scheme	100%	0%	-	-
1.1b	– Static WTA circuit – Temporal encoding scheme	23.00%	5.34%	-	-
1.1c	– Static WTA circuit – Temporal encoding scheme – 10 train cycles	28.25%	4.62%	-	-
1.2a	– Temporal WTA circuit – Static encoding scheme	100%	0%	$393/40,000$ (0.98%)	-
1.2b	– Temporal WTA circuit – Temporal encoding scheme	100%	0%	$163/400$ (40.75%)	-
1.2c	– Temporal WTA circuit – Temporal encoding scheme – Neuron connection weights $\boldsymbol{w}$ fixed to 1.0	100%	0%	$163/400$ (40.75%)	-

TABLE IV. Results experiment 2

ID	Custom parameters	Avg. acc	Stdv. acc	Sparsity $\mathcal{L}_1$	Sparsity $\mathcal{L}_2$
2.1	Temporal WTA circuit	26.25%	6.05%	$400/400$ (99.95%)	-
2.2	– Two-layer temporal WTA network – $\mathcal{L}_1$ uses softmax neurons – $K_1 = 4$	100%	0%	$156/400$ (39.00%)	$4/64$ (6.25%)
2.3a	– Two-layer temporal WTA network – $\mathcal{L}_1$ uses stochastic neurons – $K_1 = 4$	54.25%	13.60%	$121/400$ (30.20%)	$2/64$ (3.75%)
2.3b	– Two-layer temporal WTA network – $\mathcal{L}_1$ uses stochastic neurons – $K_1 = 16$	90.25%	7.45%	$208/1,600$ (13.03%)	$13/1,024$ (1.25%)
2.3c	– Two-layer temporal WTA network – $\mathcal{L}_1$ uses stochastic neurons – $K_1 = 16$ – $\mathcal{N}_1 = 5$	100%	0%	$209/1,600$ (13.05%)	$319/25,600$ (1.25%)

TABLE V. Results experiment 3

ID	Custom parameters	Avg. acc	Stdv. acc	Sparsity $\mathcal{L}_1$	Sparsity $\mathcal{L}_2$
3.1a	1 train cycle	83.39%	5.97%	$310,013/4,326,400$ (7.17%)	$76,389/5,500,000$ (1.39%)
3.1b	5 train cycles	99.98%	0.14%	$392,205/4,326,400$ (9.07%)	$45,359/5,500,000$ (0.82%)
3.1c	10 train cycles (default)	99.99%	0.07%	$358,309/4,326,400$ (8.28%)	$35,271/5,500,000$ (0.64%)
3.2a	$\mathcal{N}_1 = 1$	98.70%	2.09%	$355,806/4,326,400$ (8.22%)	$1,298/220,000$ (0.59%)
3.2b	$\mathcal{N}_1 = 10$	100%	0%	$358,903/4,326,400$ (8.30%)	$143,250/22,000,000$ (0.65%)
3.3a	$K_1 = 10$	81.70%	10.54%	$79,791/432,640$ (18.44%)	$12,813/55,000$ (23.30%)
3.3b	$K_1 = 50$	99.64%	1.01%	$244,921/2,163,200$ (11.32%)	$26,597/1,375,000$ (1.93%)
3.3c	$K_2 = 50$	99.72%	0.25%	$358,309/4,326,400$ (8.28%)	$80,288/12,500,000$ (0.64%)

TABLE VI. Results experiment 4

ID	Custom parameters	Avg. acc	Stdv. acc	Sparsity $\mathcal{L}_1$	Sparsity $\mathcal{L}_2$
4.1a	1 train cycle	72.52%	7.50%	$310,013/4,326,400$ (7.17%)	$29,465/2,750,000$ (1.07%)
4.1b	5 train cycles	88.40%	5.07%	$392,205/4,326,400$ (9.07%)	$23,677/2,750,000$ (0.86%)
4.1c	10 train cycles	88.18%	6.33%	$358,309/4,326,400$ (8.28%)	$19,478/2,750,000$ (0.71%)
4.1d	30 train cycles (default)	88.64%	4.98%	$330,983/4,326,400$ (8.28%)	$17,687/2,750,000$ (0.64%)
4.2a	$\mathcal{N}_1 = 1$	87.92%	6.96%	$330,858/4,326,400$ (7.65%)	$661/110,000$ (0.60%)
4.2b	$\mathcal{N}_1 = 10$	87.83%	5.23%	$330,888/4,326,400$ (7.65%)	$70,308/11,000,000$ (0.64%)
4.3a	$K_1 = 10$	70.45%	6.04%	$80,050/432,640$ (18.50%)	$6,148/27,500$ (22.36%)
4.3b	$K_1 = 50$	87.82%	5.61%	$229,072/2,163,200$ (10.59%)	$12,648/687,500$ (1.84%)
4.3c	$K_1 = 150$	86.92%	5.31%	$401,287/6,489,600$ (6.18%)	$24,102/6,187,500$ (0.39%)

TABLE VII. Results experiment 5

ID	Custom parameters	Avg. acc	Stdv. acc	Sparsity $\mathcal{L}_1$	Sparsity $\mathcal{L}_2$
5.1a	0 train cycles (i.e., no training)	9.79%	0.27%	$4,326,400/4,326,400$ (100%)	$25,000,000/25,000,000$ (100%)
5.1b	3 train cycles	60.43%	3.16%	$938,109/4,326,400$ (21.68%)	$180,113/25,000,000$ (0.72%)
5.1c	5 train cycles (default)	60.88%	1.00%	$954,555/4,326,400$ (22.06%)	$171,334/25,000,000$ (0.69%)
5.1d	7 train cycles	62.26%	1.74%	$963,172/4,326,400$ (22.26%)	$166,864/25,000,000$ (0.67%)
5.2a	$\mathcal{N}_1 = 1$	39.82%	0.32%	$954,095/4,326,400$ (22.05%)	$6,860/1,000,000$ (0.69%)
5.2b	$\mathcal{N}_1 = 10$	63.72%	1.18%	$954,254/4,326,400$ (22.06%)	$683,832/100,000,000$ (0.68%)
5.3a	– $K_1 = 50$ – $K_2 = 50$	43.44%	1.16%	$517,117/2,163,200$ (23.91%)	$64,461/3,125,000$ (2.06%)
5.3b	– $K_1 = 150$ – $K_2 = 150$	66.87%	0.63%	$1,367,403/6,489,600$ (21.07%)	$299,158/84,375,000$ (0.35%)
5.4a	$\hat{\eta} = 0.80$	55.50%	1.59%	$1,026,725/4,326,400$ (23.73%)	$165,304/25,000,000$ (0.66%)
5.4b	$\hat{\eta} = 1.00$	44.13%	0.55%	$1,092,333/4,326,400$ (25.25%)	$158,765/25,000,000$ (0.64%)
5.5a	Maximum of 1 simultaneous spike in layer $\mathcal{L}_1$	56.63%	1.11%	$906,402/4,326,400$ (20.95%)	$47,800/25,000,000$ (0.19%)
5.5b	Maximum of 5 simultaneous spike in layer $\mathcal{L}_1$	55.54%	0.86%	$972,760/4,326,400$ (22.48%)	$255,668/25,000,000$ (1.02%)
5.6a	– MFCC deltas are omitted from input – $\mu^{max} = 500$ (to account for less input)	32.63%	2.46%	$259,266/1,081,600$ (23.97%)	$155,762/25,000,000$ (0.62%)
5.6b	– MFCCs are omitted from input – $\mu^{max} = 500$ (to account for less input)	50.23%	2.76%	$230,994/1,081,600$ (21.36%)	$150,015/25,000,000$ (0.60%)
5.6c	$B = 6$	59.17%	1.26%	$670,506/2,433,600$ (21.36%)	$353,027/25,000,000$ (1.41%)
5.6c	$B = 10$	55.56%	0.61%	$1,238,378/6,760,000$ (18.32%)	$96,961/25,000,000$ (0.39%)
5.7a	The data is shuffled	58.05%	1.60%	$1,005,863/4,326,400$ (23.25%)	$162,452/25,000,000$ (0.65%)
5.7b	Neuron connections weights $\boldsymbol{w}$ are unfixed	54.19%	0.66%	$957,253/4,326,400$ (23.25%)	$185,568/25,000,000$ (0.74%)
5.7c	– 7 train cycles – $\mathcal{N}_1 = 10$ – $K_1 = 150$ – $K_2 = 150$	72.67%	1.99%	$1,386,243/6,489,600$ (21.26%)	$1,176,268/337,500,00$ (0.35%)